Local Advantage Networks for Multi-Agent Reinforcement Learning in Dec-POMDPs

Raphaël Avalos *raphael.avalos@vub.be*
Vrije Universiteit Brussel

Mathieu Reymond *mathieu.reymond@vub.be*
Vrije Universiteit Brussel

Ann Nowé *ann.nowe@vub.be*
Vrije Universiteit Brussel

Diederik M. Roijers *diederik.roijers@vub.be*
Vrije Universiteit Brussel
City of Amsterdam

Reviewed on OpenReview: *https://openreview.net/forum?id=adpKzWQunW*

Abstract

Many recent successful off-policy multi-agent reinforcement learning (MARL) algorithms for cooperative partially observable environments focus on finding factorized value functions, leading to convoluted network structures. Building on the structure of independent Q-learners, our LAN algorithm takes a radically different approach, leveraging a dueling architecture to learn for each agent a decentralized best-response policies via individual advantage functions. The learning is stabilized by a centralized critic whose primary objective is to reduce the moving target problem of the individual advantages. The critic, whose network's size is independent of the number of agents, is cast aside after learning. Evaluation on the StarCraft II multi-agent challenge benchmark shows that LAN reaches state-of-the-art performance and is highly scalable with respect to the number of agents, opening up a promising alternative direction for MARL research.

1 Introduction

Reinforcement learning (RL) (Sutton & Barto, 1998) is the branch of machine learning dedicated to learning through trial-and-evaluation by interaction between an agent and an environment. Research in RL has successfully managed to exceed human performance in many tasks including Atari games (Mnih et al., 2015) and the challenging game of Go (Silver et al., 2016).

While single-agent RL has been highly successful, many real world tasks – sensor networks (Mihaylov et al., 2010), wildlife protection (Xu et al., 2020), and space debris cleaning (Klima et al., 2018) – require multiple agents. When these agents need to act on local observations, or the problem is too large to centralize due to the exponential growth of the joint action space in the number of agents, an explicitly multi-agent approach is required. As such, *Multi-Agent Reinforcement Learning (MARL)* (Buşoniu et al., 2008; Hernandez-Leal et al., 2019; Shoham et al., 2007) introduces additional layers of complexity over single-agent RL.

In this paper, we focus on partially observable cooperative MARL where the agents optimize the same team reward. This setting introduces two main challenges that do not exist in single-agent RL. 1) The *moving target problem* (Tuyls & Weiss, 2012): the presence of multiple learners in an environment makes it impossible for an agent to infer the conditional probability of future states. This invalidates most single-agent approaches, as the Markovian property no longer holds. 2) The *multi-agent credit assignment problem*: to learn a policy

each agent needs to determine which actions contribute to obtaining the maximum reward. While in single agent RL this problem is only temporal, as the reward can be sparse and delayed, the shared reward increases the complexity of this problem as the agents also need to determine their individual contribution.

Centralized Training with Decentralized Execution (CTDE) (Oliehoek et al., 2008a; Foerster et al., 2018; Lowe et al., 2017), has become a popular learning paradigm for MARL. The core idea behind CTDE is that even though decentralized execution is required the learning is allowed to be centralized. Specifically, during training, it is often possible to access the global state of the environment, the observations and actions of all agents allowing to break partial observability, which mitigates both the moving target problem and the credit assignment problem.

Most of the research in off-policy CTDE MARL for collaborative partially observable environments focuses on factorizing the joint Q-Value into local agent utilities such as QMIX (Rashid et al., 2018) and QPLEX (Wang et al., 2021).

In this paper, we take a radically different approach. Our *Local Advantage Networks (LAN)* algorithm learns for every agent the advantage of the best response policy to the other agents' polices. These local advantages, which are solely conditioned on the agent observation-action history, are sufficient to build a decentralized policy. In this sense, the architecture of LAN resembles independent Q-learners more than other CTDE approaches such as QMIX or QPLEX. A key element of our solution is to derive a proxy of the local Q-value that leverages CTDE to stabilize the learning of the local advantages. For each agent the Q-value proxy is composed of the sum of the local advantage with the centralized value of the joint policy. Compared to the local Q-value, LAN's proxy is able mitigate the moving target problem, by integrating the changes of the other agents' policies faster, and to reduce the multi-agent credit assignment, by learning the local advantage function for each agent. LAN is also highly scalable as the centralized value network reuses the hidden states of the local advantages to represent the joint observation-action history and the number of parameters of the centralized value does not depend on the number of agents. Finally, compared to QMIX and QPLEX which factorize the joint Q-value into individual utilities, LAN learns individual best-response Q-value proxies. This allows LAN to not have any restriction on the family of decentralized functions that it can represent, as opposed to QMIX. Indeed, in cooperative environments the optimal policies are best response policies.

We empirically evaluate LAN against independent Q-Learners (Tan, 1993; Tampuu et al., 2015) and state-of-the-art algorithms for deep MARL, i.e., VDN (Sunehag et al., 2018), QMIX and QPLEX, on the Starcraft Multi-agent Challenge (SMAC) benchmark (Samvelyan et al., 2019). We show that on the 14 maps that compose the benchmark, LAN reaches similar performance of the SOTA in 11, surpasses the others algorithms with a large margin in 2, and under-performs in 1. In the maps with the most agents, LAN's centralized network uses up to 7 times fewer parameters than QPLEX demonstrating the scalability of our algorithm. Furthermore, in two super hard maps, LAN learns a complex strategy based on an agent sacrificing itself to lure the enemies far from its teammates, showcasing LAN's capacity to mitigate the temporally extended multi-agent credit assignment problem. This strategy allows LAN to obtain a success rate of respectively 40% and 90% on two maps where the current state-of-the-art – QPLEX – struggles to obtain any wins. By improving performance on these two maps, LAN was able to achieve an average final performance on all 14 maps that is 10% better than QPLEX's score.

Importantly, the objective of this new method is not to improve performance over the SOTA but rather to present an alternative research direction to factorizing the joint Q-value.

2 Background

The setting considered in this paper are Dec-POMDPs (Oliehoek & Amato, 2016; Oliehoek et al., 2008a) $G = \langle \mathcal{A}, \mathcal{S}, \boldsymbol{\mathcal{U}}, P, R, \boldsymbol{\mathcal{O}}, O, \gamma \rangle$. At each time-step, every agent $a \in \mathcal{A}$ selects an action $u_a \in \mathcal{U}_a$ to form the joint action $\mathbf{u} \in \boldsymbol{\mathcal{U}}$, where $\boldsymbol{\mathcal{U}} = \times_a \mathcal{U}_a$, that is processed by the environment to produce: a unique reward r common to all agents; the next state $s' \in \mathcal{S}$; and the agents' joint observation $\boldsymbol{o} \in \boldsymbol{\mathcal{O}}$, where $\boldsymbol{\mathcal{O}} = \times \mathcal{O}_a$, with $o_a \in \mathcal{O}_a$ the observation of agent a. $\gamma \in [0, 1)$ is the discount factor. As the agents cannot access the real state of the environment they condition their policy on their observation-action history $\tau_a \in \mathcal{T}_a \equiv (\mathcal{O}_a, \mathcal{U}_a)^*$,

with $\boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}$, where $\boldsymbol{\mathcal{T}} = \times_a \mathcal{T}_a$ being the joint observation-action history. We refer to the observation-action history of an agent as its history, and the joint observation-action history as the joint history. To simplify the notations in this paper we assume that the observation function is deterministic. However the extension to stochastic observations is straightforward. With that setting, the next joint history $\boldsymbol{\tau}'$ is defined entirely by the current joint history, the joint action and the state $\langle \boldsymbol{\tau}, \boldsymbol{u}, s' \rangle$. The value, Q-value and advantage functions of the joint policy $\boldsymbol{\pi}$, which can be centralized or decentralized, are defined as:

$$V^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}) = \sum_{\boldsymbol{u}} \boldsymbol{\pi}(\boldsymbol{u}|\boldsymbol{\tau}) \big[R(s, \boldsymbol{u}) + \gamma \sum_{s'} P(s'|s, \boldsymbol{u}) V^{\boldsymbol{\pi}}(s', \boldsymbol{\tau}') \big]$$

$$Q^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}, \boldsymbol{u}) = R(s, \boldsymbol{u}) + \gamma \sum_{s'} P(s'|s, \boldsymbol{u}) V^{\boldsymbol{\pi}}(s', \boldsymbol{\tau}') \quad A^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}, \boldsymbol{u}) = Q^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}, \boldsymbol{u}) - V^{\boldsymbol{\pi}}(s, \boldsymbol{\tau})$$

We note that, if there is only a single agent a Dec-POMDP is a POMDP, and if this agent can observe the full state s the POMDP is an MDP.

DQN (Mnih et al., 2013) is a popular algorithm for MDPs that learns an approximation of $Q^* = \max_\pi Q^\pi$ with a neural network parametrized by θ. This θ is learned through gradient descent by minimizing $Q(s, u \mid \theta) - y^{DQN})^2$ with $y^{DQN} = r + \gamma \max_{u'} Q(s', u' \mid \theta)$. DQN uses a replay buffer to improve sample efficiency and to stabilize the learning. Dueling DQN (Wang et al., 2016) is a variant of DQN that learns both the value and the advantage, to then produce the Q-value as the sum of both instead of learning Q directly. This alternative architecture is motivated by the fact that having one part of the neural network that learns the general value of the state, and a second part that learns the effects of the actions - represented by the advantage - can be easier than learning both in the same network. DRQN uses a Recurrent Neural Network (RNN), such as a Gated Recurrent Network (GRU) (Cho et al., 2014) or an LSTM (Hochreiter & Schmidhuber, 1997), to extend DQN to partial observablity (POMDP).

3 Related work

Applying single agent RL algorithms to Dec-POMDPs, such as Independent Q-Learners (IQL) and Independent Actor-Critic, results in poor performance due to the moving target and multi-agent credit assignment problems (Tan, 1993; Tampuu et al., 2015; Foerster et al., 2018) – with the exception of stateless normal form games (Nowé et al., 2012). The replay buffer, fundamental to DQN, worsens the moving target problem as the sampled transitions are quickly outdated and off-environment as the policies evolve. Indeed, as all the agents are learning, states of transitions saved in the replay buffer might no longer be achievable by changing the policy of one agent. As removing the replay buffer does not lead to good polices, alternatives such as importance sampling and the use of fingerprints have been explored leading to small improvements (Foerster et al., 2017). In contrast, LAN's centralized value function mitigates the moving target problem sufficiently, which enables it to take advantage of the replay buffer and to reach state-of-the-art performance.

COMA (Foerster et al., 2018) and MADDPG (Lowe et al., 2017) introduced CTDE to Deep MARL by building on single-agent actor-critic algorithms but replacing the local critic with a centralized one to improve the quality of the value estimation guiding the updates. In comparison, our method, LAN, is a value-based algorithm making it more sample-efficient. While LAN's joint value is also a centralized critic, it plays an intrinsically different role, as it fosters learning coordination between the local advantage functions.

Centralized Q-Learning (CQL) and Independent Q-Learners (IQL) form the two extremes of value-based methods for MARL. On the one hand, CQL learns a unique Q-Value conditioned on the full joint action space and the joint history. While in this setting the optimal performance is better or equal to the decentralized one due to the reduction of partial observability, the agents are no longer autonomous as they rely on a central entity for execution. In addition, this algorithm does not scale well due to the exponential increase of the joint action space in the number of agents. On the other hand, IQL learns for each agent a local Q-Value conditioned on its local observation-action history. This algorithm is heavily affected by the moving target

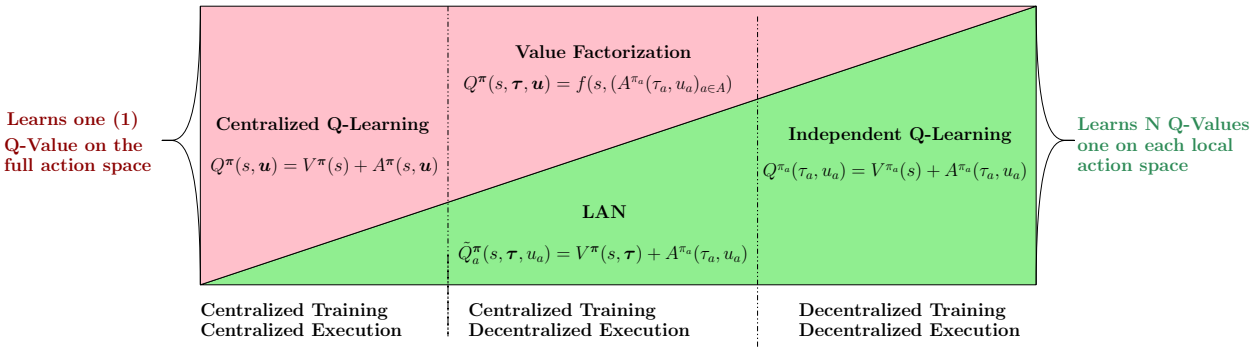


Figure 1: Comparison diagram between Centralized Q-Learning, Independent Q-Learning, Value factorization methods and LAN.

problem. However, in settings with limited interactions between agents, the moving target problem is not as intense and IQL can show good performance.

Value factorization (VDN, QMIX, QPLEX) emerged as the main alternative in recent years. Closer to CQL than IQL, those algorithms learn a unique Q-Value over the joint action space. Its factorized architecture allows recovering for each agent a utility function for action selection. To ensure that the agents select the same action during training with the centralized component and during decentralized execution, the factorization follows the individual global max (IGM) principle: the maximizing joint action of the joint Q-value must be equal to the joint action that results from maximizing the local utilities. The factorization usually enforces a monotonicity constraint to ensure IGM, i.e., for each agent the derivative of the joint Q-value to the agent's local utility is positive. VDN is the first algorithm of this kind and decomposes the joint Q-value into a simple sum. QMIX extends VDN by learning state-dependent positive weights. The state dependency broadens the family of Q-value functions that can be learned, and the positive weights constraint ensures IGM. While QMIX achieves good performance and improves over VDN, the monotonicity constraints still limits the family of functions learnable. QATTEN (Yang et al., 2020) extends QMIX by using multi-head attention (Vaswani et al., 2017) to compute the mixing weights. More recently, QPLEX extends QATTEN by transferring the IGM principle from the Q-value to the advantage function. At the cost of twice as many parameters on average and a more complex mixing network, QPLEX outperforms QMIX on SMAC. In contrast to those algorithms, LAN does not factorize the joint Q-value into individual agents utilities but learns an individual Q-value proxy for every agent. This result in LAN's architecture being able to represent any decentralized policy, as opposed to QMIX and VDN.

Figure 1 presents a visual comparison of the structural differences between CQL, IQL, value factorization and LAN. This figure highlights the fact that while value factorization and LAN are both CTDE methods, LAN is closer to IQL as it learns for each agent a Q-value on its local action space.

Improving multi-agent exploration or scalability regarding the action space in Dec-POMDPs have been successfully explored by MAVEN (Mahajan et al., 2019) and RODE (Wang & Dong, 2020). Both works are orthogonal to ours, and while they use QMIX as a base algorithm they could also be applied to LAN. For this reason we do not include them as baselines.

Recently, MAPPO (Yu et al., 2021) and IPPO de Witt et al. (2020) showed that actor-critic-based algorithms could achieve good performance on cooperative MARL. However, they require significantly more interactions, 10 million timesteps instead of 2 million, and more computing power. Comparison with those two algorithms is also harder because MAPPO changed the state space, IPPO changed the difficulty of the enemy team, and they do not use the same version of the environment. Also, they both have different hyperparameters per map whereas the other algorithms have one set of hyperparameters for the full benchmark challenge. However, just like LAN, both MAPPO and IPPO propose an alternative to off-policy value factorization. While comparing the three methods might not be straightforward due to the need to retune the three algorithms on a fixed version of SMAC, a further comparative study would help to better understand the strengths and weaknesses of each algorithm.

4 Method

In this section, we present **Local Advantage Networks (LAN)**, a novel value-based algorithm for collaborative partially observable MARL. LAN goes in the opposite direction of the current state-of-the-art in MARL, which focuses on factorizing the Q-value of the joint policy $Q^{\boldsymbol{\pi}}$ into individual utilities. Instead, LAN learns for each agent the advantage of the best response policy to the other agents' policies. The local advantages are only conditioned on the own agent's history allowing for decentralized execution. The main contribution of LAN is to stabilize the learning of those advantages by leveraging CTDE to use the value of the joint policy $V^{\boldsymbol{\pi}}$ to coordinate their learning. The centralized nature of $V^{\boldsymbol{\pi}}$ allows to reduce the partial observability, mitigate the moving target problem and the multi-agent credit assignment problem. By combining the local advantages with the centralized value, LAN derives a proxy of the individual Q-value of each agent and can simultaneously learn all components with DQN. Two key differences with a factorized Q-function are: (1) that LAN does not learn the Q-value of the joint policy, which is in fact more difficult to learn than the value V, and its factorization, but proxies of the individual Q-Values and (2) that in contrast to VDN and QMIX, LAN's architecture does not limit the the family of decentralized policies it can represent. We note that QPLEX can also represent all these policies at the cost of a more complex architecture.

Best response policies We start from the observation that in a Dec-POMDP when the agents reach an optimal policy, their individual policies are best responses to the other agents' policies. Indeed, if one agent could improve its policy while the other agents polices are fixed, the joint policy cannot be optimal as the agents share the same reward. Based on this observation, LAN focuses on learning best response polices.

To better understand how to learn best response policies, we first focus on a single agent $a \in \mathcal{A}$ and assume that the joint policy of the other agents $\boldsymbol{\pi}_{-a}$ is fixed. As in (Foerster et al., 2017), we derive from the Dec-POMDP G a POMDP $G_a = \langle \tilde{\mathcal{S}}, \mathcal{U}_a, P_a, \mathcal{O}_a, O_a, R_a, \gamma \rangle$, with $\tilde{S} = \langle \mathcal{S}, \boldsymbol{\mathcal{T}}_{-a} \rangle$ being the original state space extended with the observation-action histories of the other agents, P_a and R_a are defined as follows:

$$P_a(\langle s', \boldsymbol{\tau}'_{-a} \rangle | \langle s, \boldsymbol{\tau}_{-a} \rangle, u_a) = \sum_{\boldsymbol{u}_{-a}} \boldsymbol{\pi}_{-a}(\boldsymbol{u}_{-a}|\boldsymbol{\tau}_{-a}) P(s'|s, \langle u_a, \boldsymbol{u}_{-a} \rangle) p(\boldsymbol{\tau}'_{-a}|\boldsymbol{\tau}_{-a}, s, s', \boldsymbol{u}_{-a})$$

$$R_a(\langle s, \boldsymbol{\tau}_{-a}, u_a) = \sum_{\boldsymbol{u}_{-a}} \boldsymbol{\pi}_{-a}(\boldsymbol{u}_{-a}|\boldsymbol{\tau}_{-\boldsymbol{a}}) R(s, \langle u_a, \boldsymbol{u}_{-a} \rangle)$$

The value, Q-value and advantage of G_a can then be derived as follows, with $p(\tilde{s}|\tau_a)$ the probability of being in an extended state $\tilde{s} \in \tilde{\mathcal{S}}$ when τ_a is agent a's local history.

$$V^{\pi_a}(\tau_a) = \sum_{u_a} \pi_a(u_a|\tau_a) \sum_{\tilde{s}} p(\tilde{s}|\tau_a) \sum_{\boldsymbol{u}_{-a}} \boldsymbol{\pi}_{-a}(\boldsymbol{u}_{-a}|\boldsymbol{\tau}_{-a}) \big[R(s, (u_a, \boldsymbol{u}_{-a})) + \gamma \sum_{s'} P(s'|s, \langle u_a, \boldsymbol{u}_{-a} \rangle) V^{\pi_a}(\tau'_a) \big]$$

$$Q^{\pi_a}(\tau_a, u_a) = \sum_{\tilde{s}} p(\tilde{s}|\tau_a) \sum_{\boldsymbol{u}_{-a}} \boldsymbol{\pi}_{-a}(\boldsymbol{u}_{-a}|\boldsymbol{\tau}_{-a}) \big[R(s, (u_a, \boldsymbol{u}_{-a})) + \gamma \sum_{s'} P(s'|s, \langle u_a, \boldsymbol{u}_{-a} \rangle) V^{\pi_a}(\tau'_a) \big]$$

$$Q^{\pi_a}(\tau_a, u_a) = V^{\pi_a}(\tau_a) + A^{\pi_a}(\tau_a, u_a)$$

Partial observability Due to the partial observability, agent a needs to disambiguate the state of G_a corresponding to the original state s and the joint history of the other agents $\boldsymbol{\tau}_{-a}$. As the environment is no longer Markovian, the agent needs to base its policy on a belief over the extended state. The most straightforward way to compute this belief is to keep the full history of the agent. However, this strategy does not scale well in the number of time-steps or state space. As analyzed in the work on influence-based abstractions (Oliehoek et al., 2012), in a Dec-POMDP maintaining a belief over the subset of features that allows to locally regain the Markovian property is sufficient, using the property of d-separation. This belief is much more compact than keeping track of the entire action-observation history, and therefore offers the possibility to keep a fully sufficient representation that remains tractable. In the ideal case, the RNN's history representation will capture the belief over the d-separating features, enabling the reinforcement

learning agent to learn an optimal Dec-POMDP policy. In practice of course, we aim to closely approximate such a representation, but are often uncertain of its existence, or of its size if it does exist.

Applying DQN to the single-agent POMDP G_a learns, for each agent a, the best response policy to $\boldsymbol{\pi}_{-a}$, as the probability distribution over the relevant features P_a results from executing fixed policies for the other agents. A naive solution to learn good decentralized policies would therefore be to improve each agent successively. However, this approach fails if the environment requires the agents to explore simultaneously to find the optimal policy. On the other hand, optimizing Q^{π_a} for all the agents simultaneously, i.e., Independent Q-Learning (IQL) (Tan, 1993; Tampuu et al., 2015) also has key downsides. While IQL allows agents to explore together, it does not perform well in more complicated tasks due to the moving target problem as it ignores that the environment G_a perceived by agent a is shifting as $\boldsymbol{\pi}_{-a}$ evolves. So while we need agents that learn together, they need to do so in a coordinated manner.

Q-Value proxy LAN simultaneously learns best response policies and mitigates the moving target problem. These best response policies are expressed as *local advantage functions* that are solely conditioned on the agent's observation-action history, $A^{\pi_a}(\tau_a, u_a)$, allowing for decentralized execution. To coordinate the learning of those local advantage functions, following the CTDE paradigm, LAN leverages full information about the state and the other agents observation-action history at training time via a centralized value function $V^{\boldsymbol{\pi}}$. More specifically, LAN derives $\tilde{Q}_a^{\boldsymbol{\pi}}$ a proxy of the local Q-value Q^{π_a} for each agent $a \in A$.

$$\tilde{Q}_a^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}, u_a) = V^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}) + A^{\pi_a}(\tau_a, u_a) \tag{1}$$

The proxy is constructed by summing the local advantage A^{π_a} with the centralized value of the joint policy $V^{\boldsymbol{\pi}}$. While $\tilde{Q}_a^{\boldsymbol{\pi}}$ is not a real Q-value and it is conditioned on the full state and the joint history $\boldsymbol{\tau}$ it can be used to extract decentralized policies as the maximizing actions only depend on the agent's history τ_a, as shown by equation 2. We obtain this equation by remarking that for both decomposition of Q^{π_a} and $\tilde{Q}_a^{\pi}$, the local and centralized values are not conditioned by the agent's actions.

$$\arg\max_{u_a} \tilde{Q}_a^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}, u_a) = \arg\max_{u_a} A^{\pi_a}(\tau_a, u_a) = \arg\max_{u_a} Q^{\pi_a}(\tau_a, u_a) \tag{2}$$

LAN uses DQN to learn the individual Q-value proxy $\tilde{Q}_a^{\boldsymbol{\pi}}$ for all agents $a \in A$ simultaneously. This allows LAN to learn the local advantages A^{π_a} and the centralized value $V^{\boldsymbol{\pi}}$ in parallel by optimizing a unique loss, resulting in an efficient learning scheme. LAN's DQN target for agent a is defined as follows with the subscript t referring to a delayed copy of the networks to increase learning stability (van Hasselt et al., 2015). Appendix E contains the pseudo code of LAN.

$$y_a = r + \gamma \tilde{Q}_{t_a}^{\boldsymbol{\pi}}(s', \boldsymbol{\tau}', \arg\max_{u'_a} \tilde{Q}_a^{\boldsymbol{\pi}}(s', \boldsymbol{\tau}', u'_a)) = r + \gamma[V_t^{\boldsymbol{\pi}}(s', \boldsymbol{\tau}') + A_t^{\pi_a}(\tau'_a, \arg\max_{u'_a} A^{\pi_a}(\tau'_a, u'_a))] \tag{3}$$

The following Theorem, shows that our Q-value proxy is an unbiased estimator of the local Q-value it approximates.

Theorem 4.1. *For any agent $a \in \mathcal{A}$, and any realisable local history $\tau_a \in \mathcal{T}_a$, and any action $u_a \in \mathcal{U}_a$, the Q-value proxy $\tilde{Q}_a$ is an unbiased estimator of the local Q-value Q^{π_a}*

$$\mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \tilde{Q}_a(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, u_a) = Q^{\pi_a}(\tau_a, u_a) \tag{4}$$

We prove the Theorem in Appendix F. In a nutshell, this Theorem shows that by optimizing the Q-value proxy we are optimizing in the same direction of the local Q-value.

Compared to the local Q-value Q^{π_a}, the learning of LAN's proxy $\tilde{Q}_a^{\boldsymbol{\pi}}$ has two interesting properties that help stabilize and coordinate the learning, and give an intuition on how LAN solves the task as a whole. We note that these properties result from applying DQN to LAN's Q-value proxies to all agents in parallel, and cannot be tested independently.

Property 1: $\tilde{Q}_a^{\boldsymbol{\pi}}$ mitigates the moving target problem, which results from all the agents learning at the same time. This simultaneous learning allows the agent to explore together, which is necessary to find an

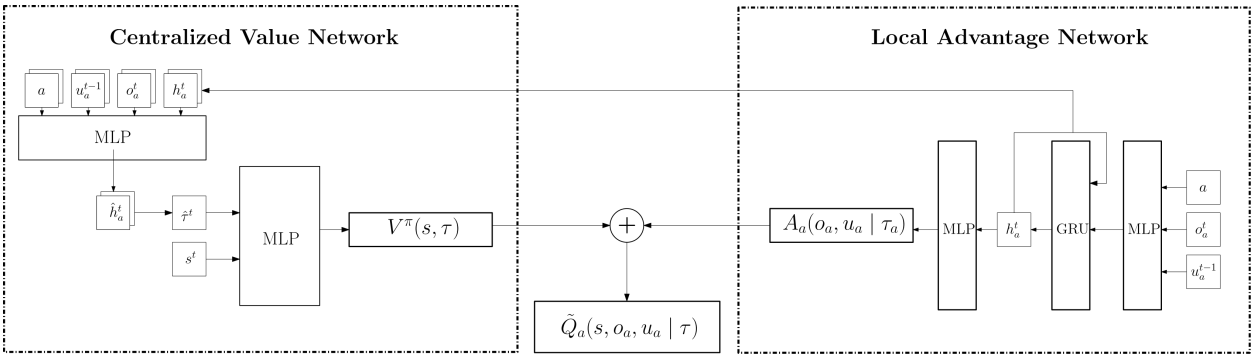


Figure 2: Architecture of LAN.

optimal strategy in non-monotonic environments, but because of it the environment is constantly changing and locally loses its Markovian property. To provide meaningful updates and prevent the learning to plateau prematurely as in IQL, the updates need to reflect as closely as possible the ever changing environment. LAN achieves this thanks to the centralized value, which coordinates the learning of all the local advantages. This happens in two steps. First, as an update of $\tilde{Q}_a^{\boldsymbol{\pi}}$ results in the update of both the centralized value and the local advantage with the same transitions, a modification of a local advantage function results in a change of the centralized value. Second, as the centralized value is part of the target update of every agent's Q-value (eq. 3), the change is then propagated to the other agents' advantage.

Property 2: $\tilde{Q}_a^{\boldsymbol{\pi}}$ mitigates the multi-agent credit assignment problem. As the centralized value function approximates the expected return of the joint policy, the agents can easily evaluate the effect of their actions on the effective return simply by subtracting it from the centralized value. This difference is learned by the local advantages. Indeed, by applying DQN to $\tilde{Q}_a^{\boldsymbol{\pi}}$ the *induced update* of the local advantage network of agent a (eq. 5) is similar to the one used by COMA (Foerster et al., 2018) to reduce the multi-agent credit assignment problem. We stress the fact that we learn all the networks in parallel with the Equation 3.

$$y_{A_a} = r + \gamma \tilde{Q}_{t_a}^{\boldsymbol{\pi}} \left(s', \boldsymbol{\tau}', \arg\max_{u'_a} \tilde{Q}_a^{\boldsymbol{\pi}} \left(s', \boldsymbol{\tau}', u'_a \right) \right) - V^{\boldsymbol{\pi}} \left(s, \boldsymbol{\tau} \right) \tag{5}$$

Additionally, we also have two intuitions regarding LAN's performance. While we were not able to prove them, we believe that they are still valuable leads to explore.

Intuition 1: $\tilde{Q}_a^{\boldsymbol{\pi}}$ allows to provide better update targets by breaking the partial observability. In a POMDP, the same observation-action history can be linked to different states forcing the agent to learn a Q-value that marginalizes over the possible states. In a Dec-POMDP this aspect is even more apparent as all the agents $a \in A$ need to marginalize over the possible states but also over the possible joint histories of the other agents $\langle s, \boldsymbol{\tau}_{-a} \rangle$ as shown by the derivation of G_a. By its conditioning on the next state and the joint history $\langle s', \boldsymbol{\tau}' \rangle$, LAN's DQN target does not suffer from the partial observability and can therefore provide updates taking into account this information. As highlighted by (Lyu et al., 2021), using a centralized target to learn a decentralized object might lead to high variance updates. The authors mention that the choice of a centralized versus decentralized critic is a bias-variance trade-off. In LAN, the value is centralized while the Advantage is decentralized. This means that LAN by using the Q-value proxy (not as precise as the real Q-values) to compute the targets induces a bias which in turn reduces the variance of the updates.

Intuition 2: $\tilde{Q}_a^{\boldsymbol{\pi}}$ reduces the learning complexity associated with decentralized policy optimization. Typically, extracting a policy from a value-based algorithm involves selecting the action that maximizes the Q-value or advantage, as they have the same action ordering. However, advantage and value functions exhibit different learning complexities, depending on the characteristics of the environment. While the advantage function learns the impact of each action on the overall return, the value function learns the expected cumulative return, necessitating more marginalization over different states and other agents' histories. This distinction motivated the introduction of Dueling DQN in MDPs (Wang et al., 2016). Nonetheless, learning the advantage function in isolation is not feasible; it requires learning the corresponding value function,

which suffers from both the partial observability and the moving target problem. Therefore, LAN's proxy provides a straightforward and efficient approach to learn local advantages without relying on local values.

Architecture

To overcome the partial observability the local advantages networks use a GRU which learns to represent the observation-actions history into a hidden state h_a, with the aim to capture the necessary features to locally regain the Markov property as stated above. This hidden state is then used to compute the local advantages. LAN leverages the work done at the agent level to represent τ_a to build a representation of $\boldsymbol{\tau}$.

For each agent a the centralized value network combines the id a of the agent with its hidden state h_a, its last observation o_a and its last action u_a into a vector $\tilde{h}_a = [h_a, o_a, u_a, a]$. To represent $\boldsymbol{\tau}$ efficiently we first embed $\tilde{h}_a$ into $\hat{h}_a$ for all agents with a shared network and sum those embedding. The embedding allows to limit the potential information loss of the summation, and this combination performs better than concatenation. Finally, the value is computed from $\boldsymbol{\tau}$ using an MLP. LAN's architecture, represented in Figure 2, provides two main benefits. First, the centralized value network does not learn a second recurrent network, which are knowingly difficult to train. Second, as the embedding for all agents are computed with the same weights, the number of parameters of the centralized value network does not depend on the number of agents.

As the policies are deterministic, the local advantages should be negative with the maximizing value equal to 0. However as (Wang et al., 2016) studies, even when computing the real Q-value in single agent MDP enforcing this constraint has a negative impact on the learning. Their experiments showed that applying the following transformation to the output of the neural network provides better stability.

$$A^{\pi_a}(\tau_a, u_a) \leftarrow A^{\pi_a}(\tau_a, u_a) - \frac{1}{|U_a|} \sum_{u \in u_a} A^{\pi_a}(\tau_a, u) \tag{6}$$

In the single agent case, this results in the learned advantage to differ from the real advantage by a fixed offset. In LAN, as the centralized value is shared between all the agents, enforcing the local advantages to have a zero mean means that the offset will be shared between all the agents. As in (Wang et al., 2016), we investigated enforcing negative advantages and observed that the learning was also highly impacted by it in LAN. While sharing the offset between the agents can have a positive impact on collaboration it can also hinder the learning by adding an additional constraint on both networks. Appendix D reports LAN's performance with the mean constraint (eq. 6). Therefore, in LAN we do not apply any constraint on the output of the advantage network.

5 Experiments

To benchmark LAN we use the StarCraft Multi-Agent Challenge[1] (SMAC) (Samvelyan et al., 2019), a set of environments that runs in the popular video game StarCraft II. SMAC does not focus on the full game but rather on micromanagement tasks where two teams of agents - possibly heterogeneous and imbalanced - fight. A match is considered won if the other team is eliminated within the time limit. The time limits differ per task. Each agent only observes its surroundings and receives a team reward proportional to the damage done to the other team plus bonuses for killing an enemy and winning. The action space of each agent consists of a move action to each cardinal direction, a no-op action, and an attack action for each enemy which is replaced by a heal action for each team member for the Medivacs units. The attack/heal action only affects units within range. As the agent's observation and action space are linearly dependent on the number of agents to perform well scalability is a key issue. SMAC also provides the real state of the environment, which we use as input for the centralized value. The benchmark is composed of 14 different maps that are designed to assess different aspects of cooperation. They are ranked into 3 categories: easy, hard, and super hard maps.

[1]We use version SC2.4.6.2.69232 and not SC2.4.10. Performances are not comparable between versions.

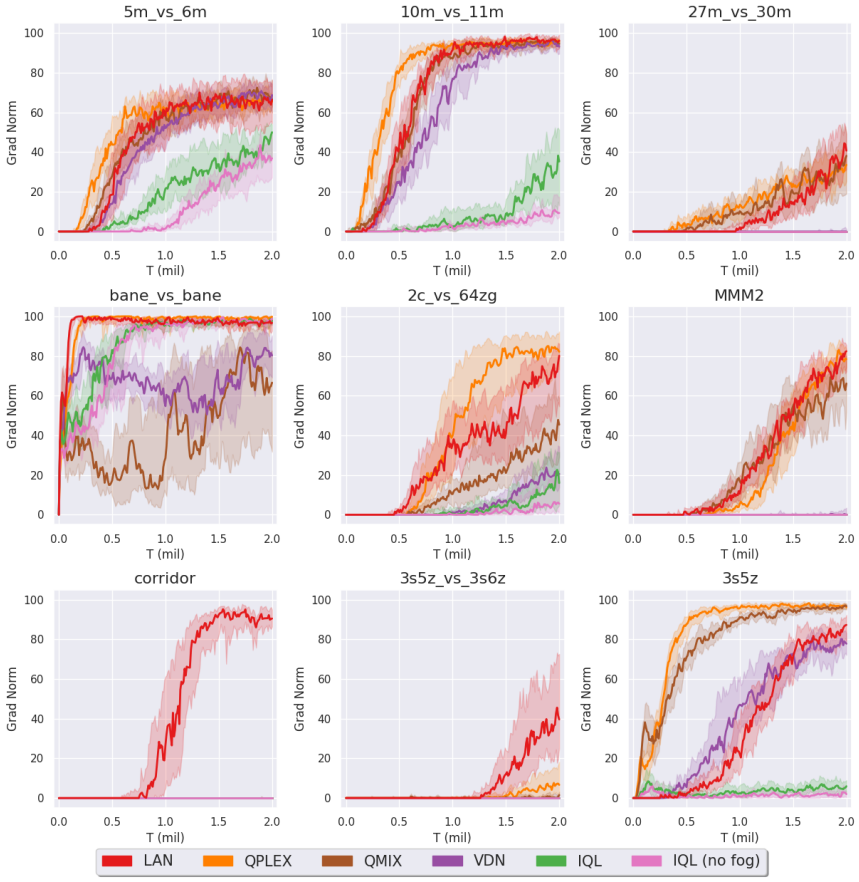


Figure 3: Median battle won rate during learning on 9 maps of SMAC. Each algorithm is run on at least 10 different seeds per map. Following the evaluation method of Samvelyan et al. (2019), we train the agents on 2 million steps and plot the median, 1st and 3rd quantiles. IQL (no fog) is introduced in 5.4

5.1 Configuration

To ensure a fair comparison, the decentralized network architecture, the version of the game, the ε-annealing parameters, the batch size, the replay buffer size, the use of a single environment, and the use of a unique set of parameters across all maps is consistent with the QMIX and QPLEX papers. Appendix B lists the hyper-parameters used, and Appendix D reports the results of a variation of LAN where we force the advantage to have a zero-mean as in Dueling DQN (Wang et al., 2016). The training and evaluation follows the procedure described in Samvelyan et al. (2019), namely 2 million training timesteps, and evaluation of the decentralized greedy polices over 32 episodes every $10k$ timesteps. We train LAN on at least 10 different random seeds and report the median of the battle win rate over the learning time as well as the first and third quantiles.

5.2 Results

We compare LAN to IQL, VDN, QMIX and QPLEX. For the first three algorithms we used the implementation of QMIX and we used the official implementation of QPLEX. In the following, we present LAN's performance on 9 maps (Figure 3). The other maps are presented in Appendix C. The first row features fights between marines with an increasing number of agents and the enemy controlling more units. The second row is composed from left to right of a balanced map with 24 heterogeneous units per team, a map where 2 power-full units fight a swarm of 64 smaller enemies, and an unbalanced heterogeneous map with a medic units that as a side effect increases the action space. The last row shows the result on two super-hard

Table 1: Number of parameters (x1000) of the value function in LAN vs. the mixing network in QPLEX/QMIX for the first 4 maps of Figure 3. See Appendix A for the other maps. The dependency of the dimension of the observation and action space in the number of agents is the only cause of the difference in the number of parameters of LAN's centralized value network in the different maps

	5m_vs_6m	**10m_vs_11m**	**27m_vs_30m**	**bane_vs_bane**
LAN	56	68	111	125
QPLEX	43	106	709	555
QMIX	32	70	283	241

maps where the baselines do not reach any wins, and a map where LAN seems to under-perform. Finally, we discuss LAN's average performance across all maps (Figure 4).

In the maps of the first row of Figure 3, two unbalanced teams with homogeneous units fight against each other, with our team composed of fewer units than the enemy: in `5m_vs_6m` 5 agents fight 6 enemies, in `10m_vs_11m` 10 agents fight 11 enemies, and in `27m_vs_30m` 27 agents fight 30 enemies. The ratio between the number of agents and the number of enemies makes the map `10m_vs_11m` easier compared to the other two. In the map `27m_vs_30m`, both the number of agents and the dimension of the observation and action space constitute a real challenge for MARL. In those three maps, LAN dominates IQL and performs on par with SOTA. First, as IQL is a natural ablation of LAN, we deduce from this experiment that the centralized value introduced by LAN does indeed help to coordinate the learning of the agents and that LAN can address the shortcomings of IQL. Second, LAN does not only performs on par with the SOTA, and slighty outperforms the other algorithms in the more difficult map, it is also more scalable than QMIX and QPLEX in terms of parameters of its centralized component with respect to the number of agents (Table 1). Indeed, between `5m_vs_6m` and `27m_vs_30m` the number of agents is multiplied by 5.4 and the number of parameters of LAN's centralized value is only multiplied by a factor of 2, while for the centralized component of QMIX and QPLEX this factor is respectively 8.8 and 16.5.

The second row of Figure 3, is composed of two hard and one super-hard maps. The first one, `bane_vs_bane`, opposes two large and balanced teams of 24 heterogeneous units. We observe that while IQL easily reaches 100% of winning rate, VDN struggles to learn and QMIX fails to learn. This hints at a limitation of both monotonous mixing strategies regarding scaling to a large number of agents, supporting our claim that an alternative research direction to value factorization is needed. QPLEX is able to learn the perfect strategy at the cost of doubling the number of parameters compared to QMIX. LAN also learns to consequently eliminate the opposing team and reaches a perfect score with 5 times fewer parameters than QPLEX. The second map, `2c_vs_64zg`, matches two powerful agents against 64 weaker agents. The numerous enemies make the action space very large, with 70 actions, which is a known challenge in RL (Zahavy et al., 2018). In this map, QPLEX reaches a final performance of 83% win rate followed closely by LAN with 80%, while QMIX, VDN and IQL score respectively around 50%, 20% and 15% win rate. The third map, `MMM2`, features two unbalanced heterogeneous teams, with the enemy team having 2 additional units, and is the only map including medical units. While IQL and VDN do not obtain any wins, QMIX and QPLEX score 60% and 80% respectively. LAN obtains the same final performance as QPLEX.

The last row of Figure 3 presents LAN's performance on 2 super-hard maps alongside the easier version of one of those maps. In the super hard map `corridor`, 6 agents of type 'zealot' fight a team 24 enemies of type 'zerlings'. While the SMAC paper claimed that the only solution for this map was to take advantage of the terrain (a spawning zone connected to a second zone by a corridor) to limit the number of enemies that can attack our agents, LAN discovered another solution. One agent lures part of the enemies to a remote location while the rest fights the remaining enemies. After killing the bait a fraction of the enemies attack our agents while the majority go through the corridor to reach the second zone. Our agents defeat their attackers, and after regenerating part of their shields move to the second zone to finish off the enemies. While the current SOTA flattens to zero, LAN obtains an almost perfect score with around 90% success rate. On the next super hard map, `3s5z_vs_3s6z`, LAN learns good decentralized policies with a performance at around 40%. The only other algorithm that was able to achieve any wins is QPLEX with less than 10%. The strategy is similar

as the one learned in corridor, a stalker (long-range unit) baits most of the enemy's zealots (close combat units) into targeting him. It then flees far away from his teammates and sacrifices himself so that the other agents can kill the stalkers and remaining zealots. The agents can then easily kill the remaining enemies as they are no longer protected by any long-range support. The last map of Figure 3, `3s5z`, is the balanced version of the previous map and therefore easier. In this map, LAN reaches 87% median battle win rate, whereas VDN only scores 80%, and QMIX and QPLEX obtain 97%. This underperformance is intriguing as LAN performs better than the other algorithms in the harder version of this map. By visualizing the learned policies in `3s5z` we discovered that LAN converges to two different policies: a) a basic confrontation policy which is the policy learned by QMIX and QPLEX; b) a baiting strategy identical to the one learned in `3s5z_vs_3s6z`. We also remark that LAN appears to still be learning and might converge to the same performance as the other QPLEX if given more time.

LAN's performance in last two super-hard maps can be attributed to is its ability to train an agent to lure the enemies and to sacrifice itself for the team's survival. We believe that this behavior is easier to discover with LAN than with the mixing algorithms because of the shared Value network, as it allows dead agents to benefit directly from the rewards scored by the other agents after their death. LAN, by focusing on learning best response policies instead of factorizing a joint Q-value, learns for each agent the policy that maximizes the team return. On the other hand, QMIX and QPLEX introduce individual rewards through factorization, which agents learn to maximize. However, if these individual rewards do not align with the team reward, as is the case in baiting strategies, mixing algorithms struggle to learn effectively. The complex strategy learned by LAN demonstrates its capacity to mitigate effectively the multi-agent credit assignment problem.

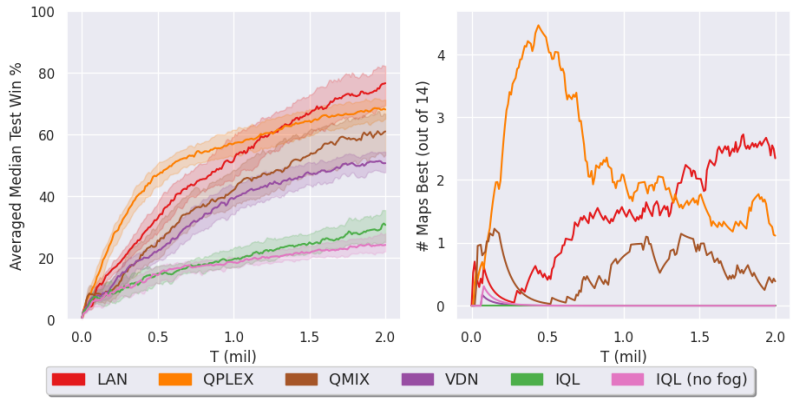


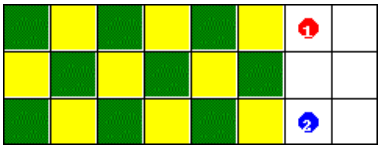
Figure 4: (Left) Averaged median test win on the 14 maps during learning. Shaded area denotes average first and third quantile. (Right) Number of maps where the algorithms are first by at least 1/32 during learning.

As in the SMAC benchmark and QPLEX papers, Figure 4 shows, on the left plot, LAN's general average performance on the 14 maps that composes the SMAC benchmark, and, on the right plot, the number of maps where each algorithm outperforms the others by a margin of at least $1/32^{\text{th}}$. IQL only achieves 30% averaged median test wins and is the best on 0 maps. This under-performance was expected as it is the only fully decentralized learning algorithm, and because it is highly vulnerable to the moving target problem. At the beginning of the learning, VDN and QMIX show similar performance, but QMIX takes the lead obtaining 60% and beating VDN by 8%. QPLEX learns faster than the other algorithms and reaches the same final performance of QMIX in just a million timesteps to obtain 67% at the end of the learning. Finally, LAN learns faster than the baselines except QPLEX, which it exceeds at around 1.25×10^6 timesteps. LAN finishes first with 77% wins. The right plot shows that LAN bests the other algorithms on 3 maps, namely `corridor`, `3s5z_vs_3s6z`, `5m_vs_6m`.

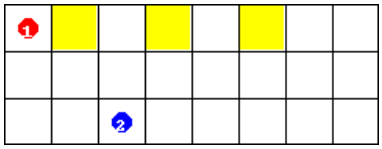
5.3 Credit assignment analysis

In the most difficult maps of SMAC the enemy teams have more units and the contribution of all the agents is required to win. The difference of performance between `3s5z` and `3s5z_vs_3s6z` (same team of agents

but one more enemy) is a good example of that. The baiting strategy discovered in `3s5z_vs_3s6z` and `corridor` showcase the credit assignment of LAN. Indeed, while the agent that serves as bait acts at the beginning of the episode the correct behavior is reinforced even though the rewards for killing the enemies and for defeating the enemy team arrives later.



(a) Initial state of the environment.



(b) Final state of the environment reached by LAN's policy, with the first agent having eaten all the apples.

Figure 5: The Checkers environment. The green boxes are the apples, they yield +10 rewards when eaten by the first agent and +1 when eaten by the second agent. The yellow boxes are the lemons that yields −10 and −1 to the first and second agent respectively. The supplementary material contains a gif of the policy.

To further emphasize this, we performed an additional experiment on Checkers, an environments of VDN designed to asses credit assignment. In Checkers the red agent gets +10 rewards for eating apples (green) and −10 rewards for eating lemons (yellow), while the second agent gets +1 and −1 respectively. The agents receive the sum of both rewards. Each agent receives as observation its location in the map and a $3x3$ window around it. The environment finishes when there are no more apples or after 100 steps. Agent 2 needs to eat the lemons (-1 reward) that block the way for agent 1 to eat the apples (+10 reward), as shown by the initial state of the environment (Figure 5a). While the agents get the same team reward, they have distinctive roles as the second agent needs to learn that negative immediate rewards lead to a better team return. LAN converges to the policies described above, with the 3 lemons on the top row left uneaten (Figure 5b). As this environment was designed to assess the credit assignment problem, this shows that LAN mitigates it.

5.4 Moving target problem analysis

IQL serves as a natural ablation of LAN, wherein the shared centralized Value component of our Q-Value proxy is swapped. As discussed in the preceding section, the primary drawback of IQL lies in its susceptibility to the moving target problem, as it disregards the learning of other agents. Consequently, IQL lacks any mitigation strategy against this issue. In scenarios such as `bane_vs_bane`, where coordination is unnecessary or when agents have no mutual influence, IQL can exhibit satisfactory performance. However, the notable superiority of LAN over IQL across all maps demonstrates that LAN effectively addresses the limitations of IQL, including the challenge posed by the moving target problem.

Since the centralized Value of LAN allows to break partial observability we carried out an additional experiment to make sure that the increased performance of LAN was not only due to targets with increased observability. In this experiment, we trained IQL without the fog of war so that all the agents could observe the entire map. While the RNN is no longer needed we kept the same architecture and training procedure of replaying full episodes. This experiment is labelled as "IQL (no fog)" in Figures 3 and 4. In all the maps IQL performs better than IQL without the fog of war. This shows that LAN's performance is not only due to the increased observability of its centralized component and strengthens our claim that our Q-Value proxy mitigates the moving target problem.

In summary, LAN performs on par with the SOTA on the easy and hard maps while dominating the other methods on the super hard maps, even the ones where the other methods did not achieve any wins. LAN outperforms QPLEX by 10% in averaged performance. These results showcase LAN's performance and scalability potential, and its capacity to handle many agents and large observation and action spaces.

6 Conclusion

In this paper, we proposed Local Advantage Networks (LAN); a novel value-based MARL algorithm for Dec-POMDPs. LAN leverages the CTDE approach by building, for each agent, a proxy of the local Q-value composed of the local advantage and the joint value. LAN trains both networks by applying DQN to a Q-value proxy. The centralized learning allows to condition the joint value on the real state to overcome the partial observability during training. In parallel, it learns the advantages together with the joint value, to synchronize all value functions to the ever changing policies. This results in more accurate DQN targets and mitigates the moving target problem. Conditioning the local advantages solely on the agent's observation-action history, ensures decentralized execution. To ensure scalability, LAN's joint value efficiently summarizes the hidden states produced by the GRUs of the local advantages to represent the joint history. Therefore, the number of parameters of this value function is independent of the number of agents.

We evaluated LAN on the challenging SMAC benchmark where we performed significantly better or on par compared to state-of-the-art methods, while its architecture is significantly more scalable in the number of agents. In the two most complex maps, LAN was able to learn a complex strategy where one agent would sacrifice itself for the survival of the team, and therefore proving experimentally LAN's ability to mitigate the multi-agent credit assignment problem. We believe that the lean architecture of LAN for learning decentralized policies in a Dec-POMDP is key to learning efficiently in decentralized partially observable settings.

Most of the recent work in value-based Deep MARL for Dec-POMDP focused on improving the value factorization of QMIX. The need for a different research direction is therefore real, and LAN, by moving away from value factorization, offers an alternative. LAN is not only able to achieve better performance than value factorization but is also more scalable parameter-wise.

Future work In future work, we aim to explore how the history representation of the centralized value can be improved through the use of Attention (Vaswani et al., 2017) or Graph Neural Networks (Kipf & Welling, 2017). We also aim to investigate how explicit communication (Oliehoek et al., 2008b; Messias et al., 2011; Wang et al., 2020; Das et al., 2019) can be added to LAN to further improve the coordination between the agents and to improve robustness of the learned policies. We also plan to investigate how LAN's architecture might benefit MARL algorithms in settings with continuous action spaces.

Acknowledgements

R. Avalos is supported by the Research Foundation – Flanders (FWO), under grant number 11F5721N. We thank Florent Delgrange and the anonymous reviewers for their valuable feedback.

References

Lucian Buşoniu, Robert Babuška, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 38(2): 156–172, 3 2008. doi: 10.1109/TSMCC.2007.913919.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pp. 1724–1734, 6 2014. URL `https://arxiv.org/abs/1406.1078v3`.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pp. 2776–2784, 10 2019. ISBN 9781510886988. URL `http://arxiv.org/abs/1810.11187`.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? 11 2020. URL http://arxiv.org/abs/2011.09533.

Jakob Foerster, Nantas Nardell, Gregory Farquhar, Trtantafyllos Afouras, Philip H.S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *34th International Conference on Machine Learning, ICML 2017*, volume 3, pp. 1879–1888. International Machine Learning Society (IMLS), 2 2017. ISBN 9781510855144. URL http://arxiv.org/abs/1702.08887.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 2974–2982, 5 2018. ISBN 9781577358008. URL http://arxiv.org/abs/1705.08926.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems 2019 33:6*, 33(6):750–797, 10 2019. ISSN 1573-7454. doi: 10.1007/S10458-019-09421-1. URL https://link.springer.com/article/10.1007/s10458-019-09421-1.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/NECO.1997.9.8.1735. URL http://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

Bojun Huang. Steady state analysis of episodic reinforcement learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/69bfa2aa2b7b139ff581a806abf0a886-Abstract.html.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 9 2017. URL http://arxiv.org/abs/1609.02907.

Richard Klima, Daan Bloembergen, Rahul Savani, Karl Tuyls, Alexander Wittig, Andrei Sapera, and Dario Izzo. Space debris removal: Learning to cooperate and the price of anarchy. *Frontiers Robotics AI*, 5 (JUN), 2018. doi: 10.3389/FROBT.2018.00054/FULL.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pp. 6380–6391, 6 2017. URL http://arxiv.org/abs/1706.02275.

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning. *Proc. of the 20th International Conference on AutonomousAgents and multi-agent Systems (AAMAS 2021)*, 2 2021. URL http://arxiv.org/abs/2102.04402.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: Multi-Agent Variational Exploration. *Advances in Neural Information Processing Systems*, 32, 10 2019. URL https://arxiv.org/abs/1910.07483v2.

João V. Messias, Matthijs T.J. Spaan, and Pedro U. Lima. Efficient offline communication policies for factored Multiagent POMDPs. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 2011.

Mihail Mihaylov, Karl Tuyls, and Ann Nowé. Decentralized Learning in Wireless Sensor Networks. In Matthew Taylor and Karl Tuyls (eds.), *Adaptive and Learning Agents*, pp. 60–73, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-11814-2.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. 12 2013. URL http://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2 2015. ISSN 14764687. doi: 10.1038/nature14236.

Ann Nowé, Peter Vrancx, and Yann Michaël De Hauwere. Game theory and multi-agent reinforcement learning. In *Adaptation, Learning, and Optimization*. 2012. doi: 10.1007/978-3-642-27645-3{_}14.

Frans A Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. 2016. ISBN 978-3-319-28927-4. URL http://www.springer.com/us/book/http://link.springer.com/10.1007/978-3-319-28929-8http://www.springer.com/us/book/%0Ahttp://link.springer.com/10.1007/978-3-319-28929-8.

Frans A. Oliehoek, Matthijs T.J. Spaan, and Nikos Vlassis. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 10 2008a. ISSN 10769757. doi: 10.1613/jair.2447. URL http://dx.doi.org/10.1613/jair.2447.

Frans A. Oliehoek, Matthijs T.J. Spaan, Shimon Whiteson, and Nikos Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 1, 2008b.

Frans A. Oliehoek, Stefan J. Witwicki, and Leslie P. Kaelbling. Influence-based abstraction for multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*, 2012. ISBN 9781577355687.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Proceedings of the35thInternational Conference on MachineLearning, Stockholm, Sweden, PMLR 80, 2018*, 3 2018. URL https://arxiv.org/abs/1803.11485v2.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 4:2186–2188, 2 2019. URL https://arxiv.org/abs/1902.04043v5.

Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 2007. ISSN 00043702. doi: 10.1016/j.artint.2006.02.006.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. 2016.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 3, pp. 2085–2087, 6 2018. ISBN 9781510868083. URL http://arxiv.org/abs/1706.05296.

R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 1998. ISSN 1045-9227. doi: 10.1109/tnn.1998.712192.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent Cooperation and Competition with Deep Reinforcement Learning. *PLoS ONE*, 12(4), 11 2015. URL http://arxiv.org/abs/1511.08779.

Ming Tan. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Machine Learning Proceedings 1993*. 1993. doi: 10.1016/b978-1-55860-307-3.50049-6.

Karl Tuyls and Gerhard Weiss. Multiagent learning: Basics, challenges, and prospects. In *AI Magazine*, 2012. doi: 10.1609/aimag.v33i3.2426.

Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 2094–2100, 9 2015. URL https://arxiv.org/abs/1509.06461v3.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pp. 5999–6009, 2017.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex Dueling Multi-Agent Q-Learning. *International Conference on Learning Representations*, 8 2021. URL https://arxiv.org/abs/2008.01062http://arxiv.org/abs/2008.01062https://openreview.net/forum?id=Rcmk0xxIQV.

Rose E. Wang, Michael Everett, and Jonathan P. How. R-MADDPG for Partially Observable Environments and Limited Communication. 2 2020. ISSN 2331-8422. URL http://arxiv.org/abs/2002.06684.

Tonghan Wang and Heng Dong. Roma: Multi-Agent reinforcement learning with emergent roles. In *37th International Conference on Machine Learning, ICML 2020*, volume PartF16814, pp. 9818–9828, 3 2020. ISBN 9781713821120. URL http://arxiv.org/abs/2003.08039.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Frcitas. Dueling Network Architectures for Deep Reinforcement Learning. In *33rd International Conference on Machine Learning, ICML 2016*, volume 4, pp. 2939–2947, 11 2016. ISBN 9781510829008. URL https://arxiv.org/abs/1511.06581.

Lily Xu, Shahrzad Gholami, Sara Mc Carthy, Bistra Dilkina, Andrew Plumptre, Milind Tambe, Rohit Singh, Mustapha Nsubuga, Joshua Mabonga, Margaret Driciru, Fred Wanyama, Aggrey Rwetsiba, Tom Okello, and Eric Enyel. Stay ahead of poachers: Illegal wildlife poaching prediction and patrol planning under uncertainty with field test evaluations (Short Version). *Proceedings - International Conference on Data Engineering*, 2020-April:1898–1901, 4 2020. doi: 10.1109/ICDE48307.2020.00198.

Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning. 2 2020. URL https://arxiv.org/abs/2002.03939v2http://arxiv.org/abs/2002.03939.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. 3 2021. URL http://arxiv.org/abs/2103.01955.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J. Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pp. 3562–3573, 9 2018. URL http://arxiv.org/abs/1809.02121.

A StarCraft Multi-Agent Challenge

The complete information about the SMAC benchmark can be found in the introductory paper (Samvelyan et al., 2019). Table 2 lists the 14 different maps of the challenge with the number of agents in each team and the number of parameters of the centralized part of LAN, QPLEX and QMIX. Table 3 lists the number of parameters of the centralized component of LAN, QMIX and QPLEX for the 14 maps.

Table 2: The different maps of SMAC.

Map Name	Ally Units	Enemy Units
2s3z	2 Stalkers & 3 Zealots	2 Stalkers & 3 Zealots
3s5z	3 Stalkers & 5 Zealots	3 Stalkers & 5 Zealots
1c3s5z	1 Colossus, 3 Stalkers & 5 Zealots	1 Colossus, 3 Stalkers & 5 Zealots
5m_vs_6m	5 Marines	6 Marines
10m_vs_11m	10 Marines	11 Marines
27m_vs_30m	27 Marines	30 Marines
3s5z_vs_3s6z	3 Stalkers & 5 Zealots	3 Stalkers & 6 Zealots
MMM2	1 Medivac, 2 Marauders & 7 Marines	1 Medivac, 3 Marauders & 8 Marines
2s_vs_1sc	2 Stalkers	1 Spine Crawler
3s_vs_5z	3 Stalkers	5 Zealots
6h_vs_8z	6 Hydralisks	8 Zealots
bane_vs_bane	20 Zerglings & 4 Banelings	20 Zerglings & 4 Banelings
2c_vs_64zg	2 Colossi	64 Zerglings
corridor	6 Zealots	24 Zerglings

Table 3: Number of parameters (x1000) of the value function in LAN vs. the mixing network in QPLEX/QMIX.

	LAN	QPLEX	QMIX
2s3z	62	50	36
3s5z	74	90	60
1c3s5z	83	113	73
5m_vs_6m	56	43	32
10m_vs_11m	68	106	70
27m_vs_30m	111	709	283
3s5z_vs_3s6z	76	95	63
MMM2	86	136	85
2s_vs_1sc	46	18	12
3s_vs_5z	54	31	22
6h_vs_8z	61	59	42
bane_vs_bane	125	555	241
2c_vs_64zg	119	116	72
corridor	79	109	69

B Implementation details

We use neural networks with ReLu activation functions, to approximate the local advantage and the centralized value. To increase the learning speed and reduce the number of parameters we share the neural network weights of the local advantages between all the agents. The input of the advantage network conditions on the agent ID so that the policy can differ per agent. The advantage network is composed of a 2 hidden

layers, a 64 units feed forward network and a 64 units GRU, which is consistent with the architecture used in the SOTA algorithms to represent the decentralized utilities (Rashid et al., 2018; Wang et al., 2021).

The centralized value network (Figure 2, left) first computes an embedding of $\tilde{h}_a$ for each agent, $\hat{h}_a$, using a feed forward network of 128 units. The agents' embeddings are then merged together by summing them resulting in a joint history embedding of fixed size. This joint history embedding is then concatenated with the real state provided by the environment to create a state-history embedding. Finally, this state-history embedding goes through an feed forward network of two hidden layers of 128 units to compute the value.

We train LAN for 2 million timesteps using a replay buffer of $5k$ episodes. During training we use an ε-greedy exploration strategy over the local advantages, with ε decaying from 1 to 0.05 over the first $50k$ timesteps. After every episode we optimize both networks twice using Adam with a learning rate of $5e^{-4}$ and without TD(λ). For each update we sample a batch of 32 episodes from the replay buffer. The DQN target are computed with a target network that is updated every 200 gradient updates. We clip to 10 the norm of the gradient.

We note that LAN does not require parameter sharing, and that each type of agent could have its own model. In that case, every agent type also needs its own embedding network to compute $\tilde{h}_a$.

C Remaining maps of SMAC

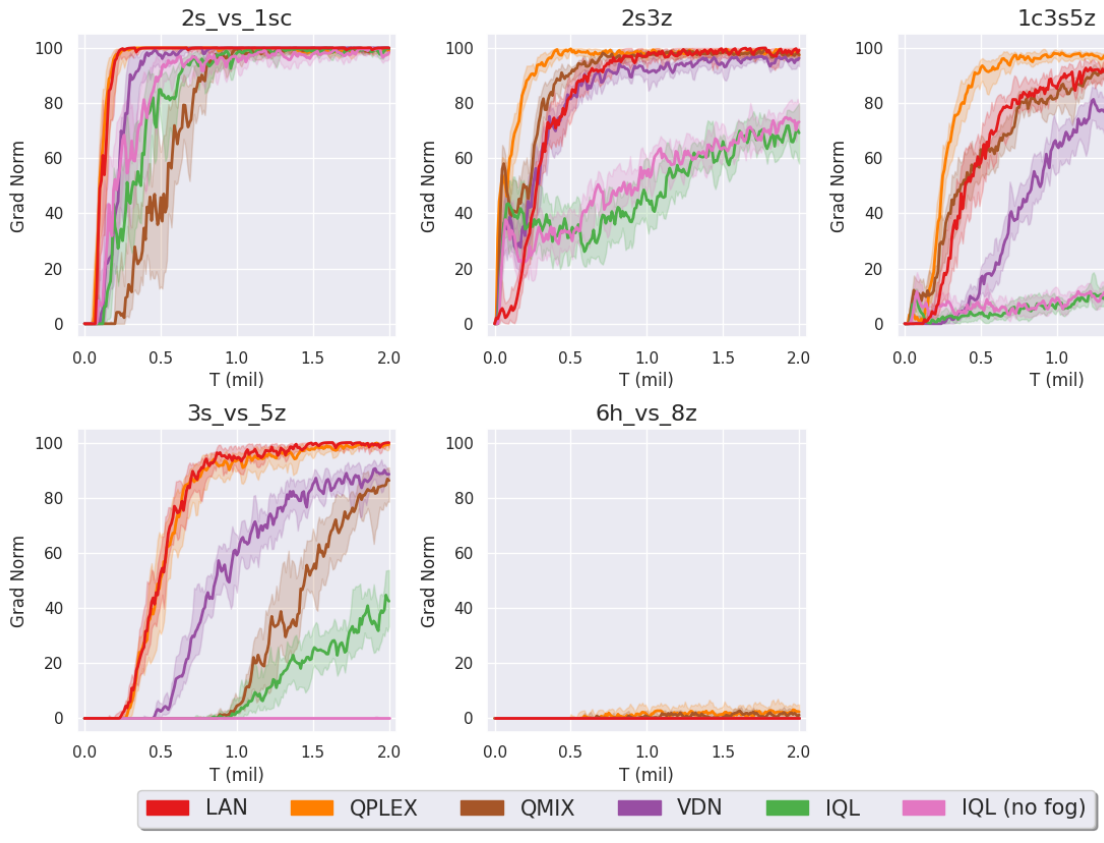


Figure 6: Median battle won rate during learning on the last 5 SMAC maps.

Figure 6 includes the 5 SMAC maps that are not included in the main paper. The first map, `2s_vs_1sc`, is an easy map and LAN learns the perfect strategy as the other algorithms do. In the second and third maps, `2s3z` and `1c3s5z`, all the algorithms but IQL learn near-optimal policies. In `3s_vs_5z`, LAN and QPLEX learn the optimal policy followed closely by QMIX and VDN that both reach around 85%. Finally,

in the last map `6h_vs_8z` no algorithm is able to score any wins. We note that the difference in performance between IQL and IQL (no fog) is consistent with the other maps: removing the fog of war does not increase performance.

D Discussion regarding the advantage

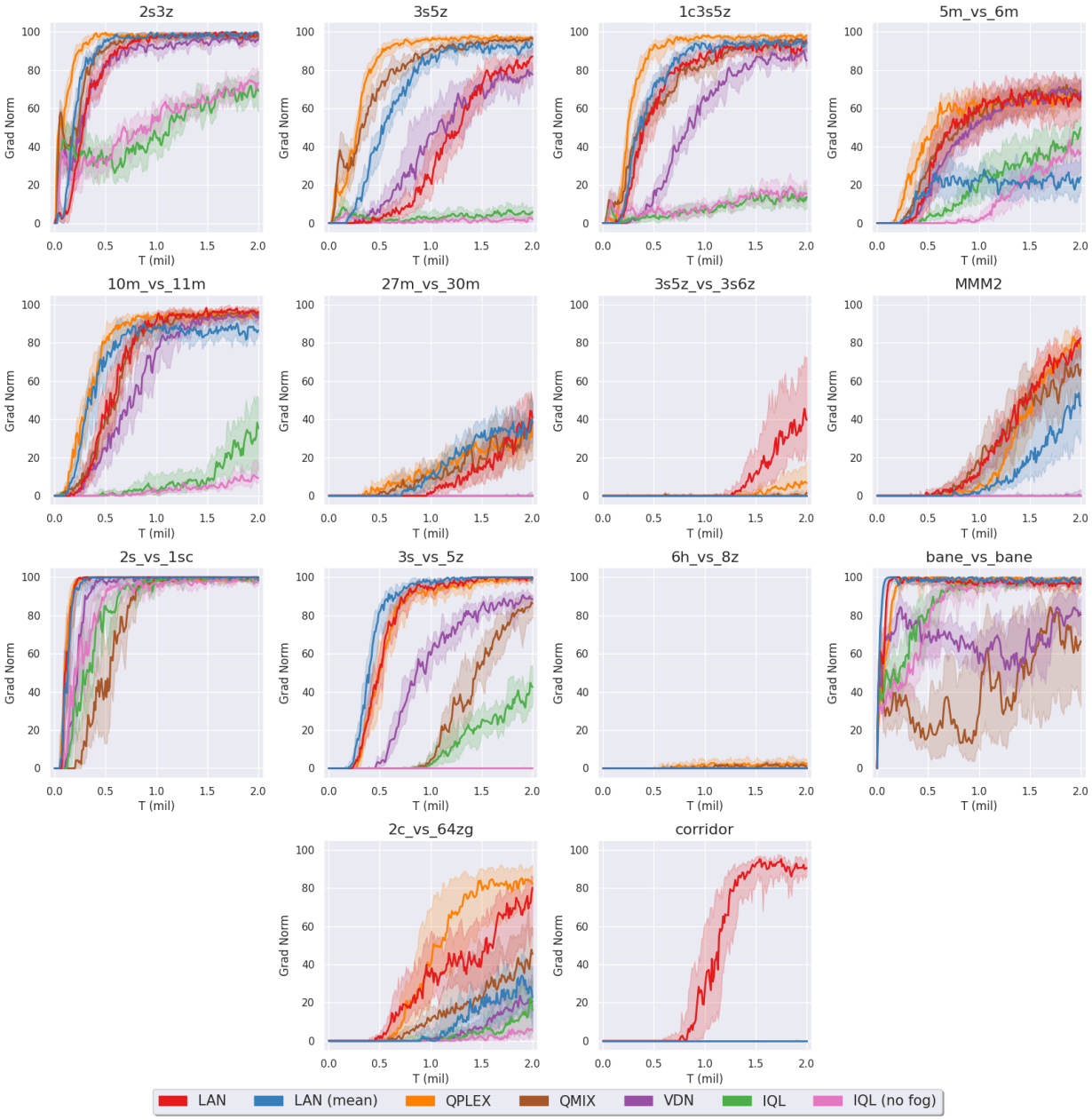


Figure 7: Median battle won rate during learning on the all the SMAC maps.

Figure 7 shows the performance on all the SMAC maps with a variation of LAN called LAN mean, which applies the equation 6 . While in two maps `3s5z`, `27m_vs_30m` the mean version of LAN improves over the classical version, it degrades the performance in others other maps such as `5m_vs_6m`, `2c_vs_64zg`, and `MMM2`, and prevents the learning in `corridor` and `3s5z_vs_3s6z`. This empirically shows that while in the single agent case the equation 6 stabilizes the learning it might not be the case when multiple agents are involved.

E Algorithm

Algorithm 1: Local Advantage Networks (LAN)

Input : Agent set N
Input : Replay memory capacity MC
Input : Frequency of target update C
Input : Exploration rate ϵ

Initialize(*replay memory D with capacity MC*);
Initialize(*centralized value function V with random weights*);
Initialize(*for each agent $a \in N$ local advantage function A^a, containing RNN_a, with random weights*);
Initialize(*target value function V_t with weights of V, and for each agent a target local advantage A^a_t with weights A^a*) ;
Initialize(*epsilon decay ϵ_{decay} and minimum epsilon $\epsilon_{\min}$*);
for *episode* **do**
    // Interaction with environment
    Initialize(*empty episode memory E*)
    ResetEnvironment($s, \boldsymbol{o} \leftarrow env$) // get state and joint observation
    ResetHiddenStates($\forall a \in N, \tau_a = 0$);
    ResetLastAction($\forall a \in N, u_a = 0$)
    while *episode is not finished* **do**
        for *agent $a \in N$* **do**
            UpdateHiddenState($\tau_a \leftarrow RNN_a(\tau_a, u_a, o_a)$);
            SelectAction($u_a \leftarrow \pi_\epsilon(A_a(\tau_a))$);
        ExecuteJointAction($\boldsymbol{u}$) ;
        Observe next state s', next joint observation $\boldsymbol{o}$, reward r ;
        StoreTransition(*s, $\boldsymbol{o}$, $\boldsymbol{u}$, r, s' $\boldsymbol{o}'$, in episode memory E*) ;
        UpdateCurrentState($s \leftarrow s'$) ;
        UpdateCurrentJointObs($\boldsymbol{o} \leftarrow \boldsymbol{o}'$) ;
    Store episode memory E in replay memory D ;
    // Perform learning step
    Sample random batch B of episodes from D ;
    for *each episode e in the batch B* **do**
        for *each timestep $t = 1$ to last step of the episode $T(e)$* **do**
            Unroll RNN of current and target networks;
            For each agent a compute current $\tilde{Q}$ estimate using Equation 1;
            // $\tilde{Q}^{\boldsymbol{\pi}}_a(s, \boldsymbol{\tau}, u_a) = V^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}) + A^{\pi_a}(\tau_a, u_a)$
            For each agent a compute TD target with target networks using Equation 3;
            // $y_a = r + \gamma[V^{\boldsymbol{\pi}}_t(s', \boldsymbol{\tau}') + A^{\pi_a}_t(\tau'_a, \arg\max_{u'_a} A^{\pi_a}(\tau'_a, u'_a))]$
            For each agent a compute $TD_{a,e,t}$ the temporal difference error;
    UpdateValueAndLocalAdvantages(*using gradient descent on the mean square temporal difference error*);
    // Update target network and exploration
    UpdateTargetNetwork($\forall a \in N, A^t_a \leftarrow A; V^t \leftarrow V$) (every C steps) ;
    UpdateExploration($\epsilon \leftarrow \max(\epsilon \times \epsilon_{decay}; \epsilon_{\min})$) ;

F Proof

Episodic process. A POMDP $\mathcal{P}$ is episodic if it includes a special *reset state* that is fully observable by the agent, and that under any policy the environment is almost surely eventually reset. Furthermore, when the environment is reset it transition to the initial state.

For this proof we consider an agent a with policy π_a and the induced POMDP G_a obtained by fixing the policy of the other agents $\boldsymbol{\pi}_{-a}$ (defined in Section 4).

Without any loss of generality, we augment G_a with an observable reset state so that G_a is episodic. This ensures the ergodicity of G_a, as every epsiodic process is ergodic or can be made ergodic without loss of generality Huang (2020), and consequently the existence of a stationary distribution $p_{\boldsymbol{\pi}}(\tilde{s}, \tau_a) = p_{\boldsymbol{\pi}}(s, \boldsymbol{\tau})$.

As LAN learns greedy policies we consider only deterministic policies.

F.1 Warm-up

By decomposing the next joint history $\boldsymbol{\tau}'$ as a tuple containing the new joint observation $\boldsymbol{o}'$, the joint action $\boldsymbol{u}$ and the joint history $\boldsymbol{\tau}$ we obtain the following equality:

$$p\big(\boldsymbol{\tau}' = \big\langle \boldsymbol{o}', \boldsymbol{u}, \tilde{\boldsymbol{\tau}} \big\rangle \mid s', \boldsymbol{\pi}(\boldsymbol{\tau}), \boldsymbol{\tau}\big) = \delta_{\tilde{\boldsymbol{\tau}}}(\boldsymbol{\tau})\delta_{\boldsymbol{u}}(\boldsymbol{\pi}(\boldsymbol{\tau}))O\big(\boldsymbol{o}' \mid s', \boldsymbol{\pi}(\boldsymbol{\tau})\big) \tag{7}$$

Where $\delta_y(x)$ is the Kronecker delta symbol. It is equal to 1 if $x = y$ and 0 otherwise.

We can obtain a similar result for the next local history τ'_a

$$p\big(\tau'_a = \big\langle o'_a, u_a, \tilde{\tau_a} \big\rangle \mid s', \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle\big) = \delta_{\tilde{\tau}_a}(\tau_a)\delta_{u_a}(\pi_a(\tau_a))O_a\big(o'_a \mid s', \pi_a(\tau_a)\big) \tag{8}$$

For any local history τ_a of agent a that is realisable under the policy π_a we can define the following conditional probability:

$$p(s, \boldsymbol{\tau}_{-a} \mid \tau_a) = \frac{p_{\boldsymbol{\pi}}(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)}{p(\tau_a)} = \frac{p_{\boldsymbol{\pi}}(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)}{\mathbb{E}_{s', \langle \boldsymbol{\tau}'_{-a}, \tau'_a \rangle \sim p_{\boldsymbol{\tau}}} \delta_{\tau'_a}(\tau_a)} \tag{9}$$

For any realisable history τ_a of agent a that is realisable under the policy π_a, and next history τ'_a we have:

$$p\big(\tau'_a \mid \tau_a\big) = \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} p\big(\tau'_a \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) \tag{10}$$

Proof:

$$
\begin{aligned}
p\big(\tau_a' \mid \tau_a\big) &= \int_s \int_{s'} \int_{\boldsymbol{\tau}_{-a}} p\big(\tau_a' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), s\big) p\big(s', \boldsymbol{\tau}_{-a}, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), s \mid \tau_a\big) \mathrm{d}\boldsymbol{\tau}_{-a} \mathrm{d}s' \mathrm{d}s \\
&\qquad \text{(law of total probability)} \\
&= \int_s \int_{s'} \int_{\boldsymbol{\tau}_{-a}} p\big(\tau_a' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), s\big) p(s, \boldsymbol{\tau}_{-a} \mid \tau_a) \mathbf{P}\big(s' \mid s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) \mathrm{d}\boldsymbol{\tau}_{-a} \mathrm{d}s' \mathrm{d}s \\
&\qquad \text{(chain rule)} \\
&= \int_s \int_{\boldsymbol{\tau}_{-a}} p(s, \boldsymbol{\tau}_{-a} \mid \tau_a) \int_{s'} \mathbf{P}\big(s' \mid s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) p\big(\tau_a' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), s\big) \mathrm{d}s' \mathrm{d}s \mathrm{d}\boldsymbol{\tau}_{-a} \\
&\qquad \text{(linearity)} \\
&= \int_s \int_{\boldsymbol{\tau}_{-a}} p(s, \boldsymbol{\tau}_{-a} \mid \tau_a) \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} p\big(\tau_a' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), s\big) \mathrm{d}s \mathrm{d}\boldsymbol{\tau}_{-a} \\
&\qquad \text{(definition of expectation)} \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} p\big(\tau_a' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), s\big) \qquad \text{(definition of expectation)} \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot \mid s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} p\big(\tau_a' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) \\
&\qquad \text{(conditional independence of } \tau_a' \text{ and } s \text{ given } s')
\end{aligned}
$$

For any realisable history τ_a, that is realisable under the policy π_a, and any next state s' and next joint history $\boldsymbol{\tau}'$ we have

$$
p\big(s', \boldsymbol{\tau}' \mid \tau_a\big) = \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \mathbf{P}\big(s' \mid s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) p\big(\boldsymbol{\tau}' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) \tag{11}
$$

Proof

$$
\begin{aligned}
p\big(s', \boldsymbol{\tau}' \mid \tau_a\big) &= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} p\big(s', \boldsymbol{\tau}' \mid s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle\big) && \text{(law of total probability)} \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} p\big(s' \mid s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle\big) p\big(\boldsymbol{\tau}' \mid s', s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle\big) && \text{(chain rule)} \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \mathbf{P}\big(s' \mid s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) p\big(\boldsymbol{\tau}' \mid s', s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \mathbf{P}\big(s' \mid s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) p\big(\boldsymbol{\tau}' \mid s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\big) \\
&\qquad \text{(conditional independence of } \boldsymbol{\tau}' \text{ and } s \text{ given } s', \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))
\end{aligned}
$$

F.2 Unbiased estimator

Theorem F.1. *For any agent $a \in \mathcal{A}$, and any realisable local history $\tau_a \in \mathcal{T}_a$, and any action $u_a \in \mathcal{U}_a$, the Q-value proxy $\tilde{Q}_a$ is an unbiased estimator of the local Q-value Q^{π_a}*

$$
\mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \tilde{Q}_a(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, u_a) = Q^{\pi_a}(\tau_a, u_a) \tag{12}
$$

Proof
We fix $a \in \mathcal{A}, u_a \in \mathcal{U}_a, \tau_a \in \mathcal{T}_a$

$$
\begin{aligned}
\mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \big[\tilde{Q}_a(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, u_a) - Q^{\pi_a}(\tau_a, u_a)\big] &= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \big[V^{\boldsymbol{\pi}}(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle) + A^{\pi_a}(\tau_a, u_a) \\
&\qquad - (V^{\pi_a}(\tau_a) + A^{\pi_a}(\tau_a, u_a))\big] \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot \mid \tau_a)} \big[V^{\boldsymbol{\pi}}(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle) - V^{\pi_a}(\tau_a)\big]
\end{aligned} \tag{13}
$$

By definition we have:

$$V^{\boldsymbol{\pi}}(s, \boldsymbol{\tau}) = r(s, \boldsymbol{\pi}(\boldsymbol{\tau})) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|s, \boldsymbol{\pi}(\boldsymbol{\tau}))} \mathop{\mathbb{E}}_{\boldsymbol{\tau}' \sim p(\cdot|s', \boldsymbol{\pi}(\boldsymbol{\tau}), \boldsymbol{\tau})} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{\tau}'\right)$$

$$V^{\pi_a}(\tau_a) = \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} r(s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)) + \gamma \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} \mathop{\mathbb{E}}_{\tau'_a \sim p(\cdot|s', \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)} V^{\pi_a}\left(\tau'_a\right)$$

We define Δ_r and Δ_p as follow:

$$\begin{aligned}
\Delta_r(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle) &= r(s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)) - \mathop{\mathbb{E}}_{\tilde{s}, \tilde{\boldsymbol{\tau}}_{-a} \sim p(\cdot|\tau_a)} r(\tilde{s}, \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle)) \\
\Delta_p(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle) &= \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} \mathop{\mathbb{E}}_{\boldsymbol{\tau}' \sim p(\cdot|s', \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{\tau}'\right) \\
&\quad - \mathop{\mathbb{E}}_{\tilde{s}, \tilde{\boldsymbol{\tau}}_{-a} \sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|\tilde{s}, \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle))} \mathop{\mathbb{E}}_{\tau'_a \sim p(\cdot|\tilde{s}', \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle), \langle \tilde{\boldsymbol{\tau}_{-a}}, \tau_a \rangle)} V^{\pi_a}\left(\tau'_a\right)
\end{aligned}$$

This allows us to rewrite Eq 13 as

$$\mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \left[\tilde{Q}_a(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle, u_a) - Q^{\pi_a}(\tau_a, u_a)\right] = \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \left[\Delta_r(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\right] + \gamma \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \left[\Delta_p(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\right] \tag{14}$$

Let's first focus on the first part of the RHS of Equation 13.

$$\begin{aligned}
\mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \left[\Delta_r(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\right] &= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \left[r(s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)) - \mathop{\mathbb{E}}_{\tilde{s}, \tilde{\boldsymbol{\tau}}_{-a} \sim p(\cdot|\tau_a)} r(\tilde{s}, \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle))\right] \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} r(s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)) - \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{\tilde{s}, \tilde{\boldsymbol{\tau}}_{-a} \sim p(\cdot|\tau_a)} r(\tilde{s}, \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle)) \\
&\qquad \text{(linearity of expectation)} \\
&= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} r(s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)) - \mathop{\mathbb{E}}_{\tilde{s}, \tilde{\boldsymbol{\tau}}_{-a} \sim p(\cdot|\tau_a)} r(\tilde{s}, \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle)) \\
&\qquad \text{(second part does not depend on } s, \boldsymbol{\tau}_{-a}) \\
&= 0
\end{aligned}$$

Let's now focus on the second part of the RHS of Equation 13.

$$\begin{aligned}
\mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \left[\Delta_p(s, \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)\right] &= \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \Bigg[\mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} \mathop{\mathbb{E}}_{\boldsymbol{\tau}' \sim p(\cdot|s', \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{\tau}'\right) \\
&\qquad - \mathop{\mathbb{E}}_{\tilde{s}, \tilde{\boldsymbol{\tau}}_{-a} \sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|\tilde{s}, \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle))} \mathop{\mathbb{E}}_{\tau'_a \sim p(\cdot|\tilde{s}', \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle), \langle \tilde{\boldsymbol{\tau}_{-a}}, \tau_a \rangle)} V^{\pi_a}\left(\tau'_a\right) \Bigg] \\
&= \overbrace{\mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|s, \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle))} \mathop{\mathbb{E}}_{\boldsymbol{\tau}' \sim p(\cdot|s', \boldsymbol{\pi}(\langle \boldsymbol{\tau}_{-a}, \tau_a \rangle), \langle \boldsymbol{\tau}_{-a}, \tau_a \rangle)} V^{\boldsymbol{\pi}}\left(s', \boldsymbol{\tau}'\right)}^{\text{A}} \\
&\qquad \underbrace{- \mathop{\mathbb{E}}_{s, \boldsymbol{\tau}_{-a} \sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{\tilde{s}, \tilde{\boldsymbol{\tau}}_{-a} \sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s' \sim \mathbf{P}(\cdot|\tilde{s}, \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle))} \mathop{\mathbb{E}}_{\tau'_a \sim p(\cdot|\tilde{s}', \boldsymbol{\pi}(\langle \tilde{\boldsymbol{\tau}}_{-a}, \tau_a \rangle), \langle \tilde{\boldsymbol{\tau}_{-a}}, \tau_a \rangle)} V^{\pi_a}\left(\tau'_a\right)}_{\text{B}} \\
&\qquad \text{(linearity of expectation)}
\end{aligned}$$

$$
\begin{aligned}
\mathrm{A} &= \mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|s,\boldsymbol{\pi}(\langle\boldsymbol{\tau}_{-a},\tau_a\rangle))} \mathop{\mathbb{E}}_{\boldsymbol{\tau}'\sim p(\cdot|s',\boldsymbol{\pi}(\langle\boldsymbol{\tau}_{-a},\tau_a\rangle),\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)} V^{\boldsymbol{\pi}}\big(s',\boldsymbol{\tau}'\big) \\
&= \mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)} \int_{s'} \mathbf{P}\big(s' \mid s,\boldsymbol{\pi}(\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)\big) \int_{\boldsymbol{\tau}'} p\big(\boldsymbol{\tau}' \mid s',\boldsymbol{\pi}(\langle\boldsymbol{\tau}_{-a},\tau_a\rangle),\langle\boldsymbol{\tau}_{-a},\tau_a\rangle\big) V^{\boldsymbol{\pi}}\big(s',\boldsymbol{\tau}'\big)\,\mathrm{d}s'\,\mathrm{d}\boldsymbol{\tau}' \\
&\qquad\qquad \text{(definition of expectation)} \\
&= \int_{s'}\int_{\boldsymbol{\tau}'} \mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)} \mathbf{P}\big(s' \mid s,\boldsymbol{\pi}(\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)\big) p\big(\boldsymbol{\tau}' \mid s',\boldsymbol{\pi}(\langle\boldsymbol{\tau}_{-a},\tau_a\rangle),\langle\boldsymbol{\tau}_{-a},\tau_a\rangle\big) V^{\boldsymbol{\pi}}\big(s',\boldsymbol{\tau}'\big)\,\mathrm{d}s'\,\mathrm{d}\boldsymbol{\tau}' \quad \text{(linearity)} \\
&= \int_{s'}\int_{\boldsymbol{\tau}'} p\big(s',\boldsymbol{\tau}' \mid \tau_a\big) V^{\boldsymbol{\pi}}\big(s',\boldsymbol{\tau}'\big)\,\mathrm{d}s'\,\mathrm{d}\boldsymbol{\tau}' \qquad \text{(see Eq. 11)} \\
&= \mathop{\mathbb{E}}_{s',\boldsymbol{\tau}'\sim p(\cdot|\tau_a)} V^{\boldsymbol{\pi}}\big(s',\boldsymbol{\tau}'\big) \qquad \text{(definition of expectation)}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{B} &= \mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{\tau}}_{-a}\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|\tilde{s},\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle))} \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tilde{s}',\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle),\langle\tilde{\boldsymbol{\tau}_{-a}},\tau_a\rangle)} V^{\pi_a}\big(\tau'_a\big) \\
&= \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{\tau}}_{-a}\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|\tilde{s},\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle))} \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tilde{s}',\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle),\langle\tilde{\boldsymbol{\tau}_{-a}},\tau_a\rangle)} V^{\pi_a}\big(\tau'_a\big) \qquad \text{(does not depend on } s,\boldsymbol{\tau}_{-a}) \\
&= \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{\tau}}_{-a}\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|\tilde{s},\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle))} \int_{\tau'_a} p\big(\tau'_a \mid \tilde{s}',\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle),\langle\tilde{\boldsymbol{\tau}_{-a}},\tau_a\rangle\big) V^{\pi_a}\big(\tau'_a\big)\,\mathrm{d}\tau'_a \\
&\qquad\qquad \text{(definition of expectation)} \\
&= \int_{\tau'_a} \mathop{\mathbb{E}}_{\tilde{s},\tilde{\boldsymbol{\tau}}_{-a}\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s'\sim\mathbf{P}(\cdot|\tilde{s},\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle))} p\big(\tau'_a \mid \tilde{s}',\boldsymbol{\pi}(\langle\tilde{\boldsymbol{\tau}}_{-a},\tau_a\rangle),\langle\tilde{\boldsymbol{\tau}_{-a}},\tau_a\rangle\big) V^{\pi_a}\big(\tau'_a\big)\,\mathrm{d}\tau'_a \qquad \text{(linearity)} \\
&= \int_{\tau'_a} p\big(\tau'_a \mid \tau_a\big) V^{\pi_a}\big(\tau'_a\big)\,\mathrm{d}\tau'_a \qquad \text{(see Eq. 10)} \\
&= \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} V^{\pi_a}\big(\tau'_a\big) \qquad \text{(definition of expectation)}
\end{aligned}
$$

By using the value of A and B we get:

$$
\begin{aligned}
\mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)} \big[\Delta_p(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)\big] &= \mathop{\mathbb{E}}_{s',\boldsymbol{\tau}'\sim p(\cdot|\tau_a)} V^{\boldsymbol{\pi}}\big(s',\boldsymbol{\tau}'\big) - \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} V^{\pi_a}\big(\tau'_a\big) \\
&= \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s',\boldsymbol{\tau}'_{-a}\sim p(\cdot|\tau_a,\tau'_a)} V^{\boldsymbol{\pi}}\big(s',\big\langle\boldsymbol{\tau}'_{-a},\tau'_a\big\rangle\big) - \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} V^{\pi_a}\big(\tau'_a\big) \quad \text{(chain rule)} \\
&= \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s',\boldsymbol{\tau}'_{-a}\sim p(\cdot|\tau'_a)} V^{\boldsymbol{\pi}}\big(s',\big\langle\boldsymbol{\tau}'_{-a},\tau'_a\big\rangle\big) - \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} V^{\pi_a}\big(\tau'_a\big) \\
&\qquad\qquad (\tau'_a \text{ contains } \tau_a) \\
&= \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s',\boldsymbol{\tau}'_{-a}\sim p(\cdot|\tau'_a)} \big[V^{\boldsymbol{\pi}}\big(s',\big\langle\boldsymbol{\tau}'_{-a},\tau'_a\big\rangle\big) - V^{\pi_a}\big(\tau'_a\big)\big] \qquad \text{(linearity)}
\end{aligned}
$$

Therefore we obtain:

$$
\mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)} \big[V^{\boldsymbol{\pi}}(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle) - V^{\pi_a}(\tau_a)\big] = \gamma \mathop{\mathbb{E}}_{\tau'_a\sim p(\cdot|\tau_a)} \mathop{\mathbb{E}}_{s',\boldsymbol{\tau}'_{-a}\sim p(\cdot|\tau'_a)} \big[V^{\boldsymbol{\pi}}\big(s',\big\langle\boldsymbol{\tau}'_{-a},\tau'_a\big\rangle\big) - V^{\pi_a}\big(\tau'_a\big)\big] \qquad (15)
$$

By applying recursively n times Equation 15 we obtain:

$$\mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)}\left[V^{\boldsymbol{\pi}}(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)-V^{\pi_a}(\tau_a)\right] = \gamma^n \mathop{\mathbb{E}}_{\tau_a^1\sim p(\cdot|\tau_a)}\mathop{\mathbb{E}}_{\tau_a^2\sim p(\cdot|\tau_a^1)}\cdots\mathop{\mathbb{E}}_{\tau_a^n\sim p\left(\cdot|\tau_a^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{\tau}_{-a}^n\sim p(\cdot|\tau_a^n)}\left[V^{\boldsymbol{\pi}}\left(s^n,\left\langle\boldsymbol{\tau}_{-a}^n,\tau_a^n\right\rangle\right)-V^{\pi_a}(\tau_a^n)\right] \tag{16}$$

We then define $R_{\max}=\max_{s\in\mathcal{S}}\max_{\boldsymbol{u}\in\boldsymbol{\mathcal{U}}}|R(s,\boldsymbol{u})|$. This allows us to bound the difference between the centralized value and the local value:

$$\begin{aligned}\forall s\in\mathcal{S},\boldsymbol{\tau}\in\boldsymbol{\mathcal{T}},\quad |V^{\boldsymbol{\pi}}(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)-V^{\pi_a}(\tau_a)| &\leqslant |V^{\boldsymbol{\pi}}(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)|+|V^{\pi_a}(\tau_a)| && \text{(triangular inequality)}\\ &\leqslant \frac{R_{\max}}{1-\gamma}+\frac{R_{\max}}{1-\gamma} && \text{(upper-bound on the value)}\\ &\leqslant \frac{2R_{\max}}{1-\gamma}\end{aligned}$$

This allows us to bound the LHS of Equation 16:

$$\begin{aligned}&\left|\mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)}\left[V^{\boldsymbol{\pi}}(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)-V^{\pi_a}(\tau_a)\right]\right|\\ &\quad=\left|\gamma^n\mathop{\mathbb{E}}_{\tau_a^1\sim p(\cdot|\tau_a)}\mathop{\mathbb{E}}_{\tau_a^2\sim p(\cdot|\tau_a^1)}\cdots\mathop{\mathbb{E}}_{\tau_a^n\sim p\left(\cdot|\tau_a^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{\tau}_{-a}^n\sim p(\cdot|\tau_a^n)}\left[V^{\boldsymbol{\pi}}\left(s^n,\left\langle\boldsymbol{\tau}_{-a}^n,\tau_a^n\right\rangle\right)-V^{\pi_a}(\tau_a^n)\right]\right|\\ &\quad\leqslant\gamma^n\mathop{\mathbb{E}}_{\tau_a^1\sim p(\cdot|\tau_a)}\mathop{\mathbb{E}}_{\tau_a^2\sim p(\cdot|\tau_a^1)}\cdots\mathop{\mathbb{E}}_{\tau_a^n\sim p\left(\cdot|\tau_a^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{\tau}_{-a}^n\sim p(\cdot|\tau_a^n)}\left[\left|V^{\boldsymbol{\pi}}\left(s^n,\left\langle\boldsymbol{\tau}_{-a}^n,\tau_a^n\right\rangle\right)-V^{\pi_a}(\tau_a^n)\right|\right]\\ &\qquad\qquad\text{(Jensen inequality)}\\ &\quad\leqslant\gamma^n\mathop{\mathbb{E}}_{\tau_a^1\sim p(\cdot|\tau_a)}\mathop{\mathbb{E}}_{\tau_a^2\sim p(\cdot|\tau_a^1)}\cdots\mathop{\mathbb{E}}_{\tau_a^n\sim p\left(\cdot|\tau_a^{n-1}\right)}\mathop{\mathbb{E}}_{s^n,\boldsymbol{\tau}_{-a}^n\sim p(\cdot|\tau_a^n)}\left[\frac{2R_{\max}}{1-\gamma}\right]\qquad\text{(see above)}\\ &\quad\leqslant\gamma^n\frac{2R_{\max}}{1-\gamma}\end{aligned}$$

As $\gamma\in]0,1[$, when $n\to+\inf$ we obtain:

$$\left|\mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)}\left[V^{\boldsymbol{\pi}}(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle)-V^{\pi_a}(\tau_a)\right]\right|\leqslant 0$$

And finally, using Eq. 13:

$$\mathop{\mathbb{E}}_{s,\boldsymbol{\tau}_{-a}\sim p(\cdot|\tau_a)}\left[\tilde{Q}_a(s,\langle\boldsymbol{\tau}_{-a},\tau_a\rangle,u_a)-Q^{\pi_a}(\tau_a,u_a)\right]=0$$

G Additional Experiment

We conducted an evaluation of LAN and the selected baseline algorithms within a modified version of the *simple spread* environment from the Multi-Agent Particle Environment suite (MPE) Lowe et al. (2017). In the original environment, three agents are tasked to spread efficiently across three landmarks while avoiding collisions with one another. The reward structure combined two components: a) the cumulative negative distance between each landmark and its closest agent; b) penalties for collisions between agents. Both agents and landmarks are randomly spawned on the map at the beginning of an episode. Notably, we introduced partial observability into the environment, restricting agents to observe only those agents and landmarks within a fixed radius. Additionally, modifications were made to the environment, allowing for any number of agents while maintaining a constant ratio between the environment size and the agent count.

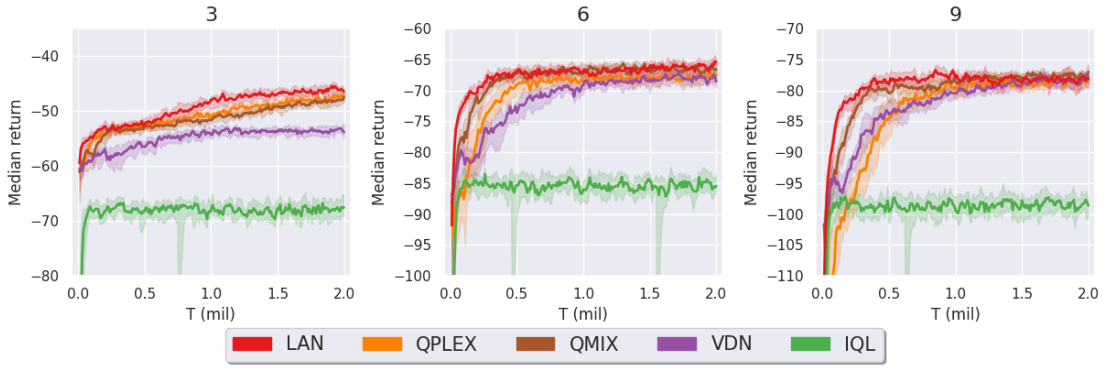


Figure 8: Median return during learning on the simple spread environment of MPE with 3, 6 and 9 agent. Each algorithm is run on 10 different seeds. We train the agents on 2 million steps and plot the median, 1st and 3rd quantiles.

Figure [8 depicts the median return obtained by LAN and the baseline algorithms with 3, 6, and 9 agents. The results are averaged over 10 runs. The hyper-parameters from SMAC were adopted without further tuning. The results consistently demonstrate LAN's accelerated learning compared to other algorithms in all three instances. With 3 agents, both QPLEX and QMIX exhibit slightly inferior performance relative to LAN. In contrast, VDN significantly underperforms in comparison to LAN, while IQL appears to struggle in learning. With 6 and 9 agents, the learning curves of LAN, QPLEX, QMIX, and VDN align closely, eventually reaching similar performance levels. However we note that LAN consistently achieves quicker convergence. IQL fails to learn a good policy in all the instances.