Off-Policy Correction For Multi-Agent Reinforcement Learning

Michał Zawalski
University of Warsaw
Warsaw, Poland
m.zawalski@uw.edu.pl

Błażej Osiński
University of Warsaw
Warsaw, Poland
b.osinski@mimuw.edu.pl

Henryk Michalewski
Google Research
henrykm@google.com

Piotr Miłoś
Polish Academy of Sciences
Warsaw, Poland
pmilos@impan.pl

ABSTRACT

Multi-agent reinforcement learning (MARL) provides a framework for problems involving multiple interacting agents. Despite apparent similarity to the single-agent case, multi-agent problems are often harder to train and analyze theoretically. In this work, we propose MA-Trace, a new on-policy actor-critic algorithm, which extends V-Trace to the MARL setting. The key advantage of our algorithm is its high scalability in a multi-worker setting. To this end, MA-Trace utilizes importance sampling as an off-policy correction method, which allows distributing the computations with no impact on the quality of training. Furthermore, our algorithm is theoretically grounded – we prove a fixed-point theorem that guarantees convergence. We evaluate the algorithm extensively on the StarCraft Multi-Agent Challenge, a standard benchmark for multi-agent algorithms. MA-Trace achieves high performance on all its tasks and exceeds state-of-the-art results on some of them.

KEYWORDS

Reinforcement Learning, V-Trace, Importance Sampling, Scalability

1 INTRODUCTION

Reinforcement learning has witnessed impressive development in recent years. Famously, superhuman performance has been achieved in games Go [30], StarCraft II [36], Dota 2 [4] and other applications. These successes are the result of rapid algorithmic development. Research in directions like trust-region optimization [29], principle of maximum entropy [13], importance sampling [8], distributional RL [3] or bridging the sim-to-real gap [2] are among these which brought significant progress. Multi-agent reinforcement learning (MARL), a framework for problems involving multiple interacting agents, is similar to the standard, single-agent setting. However, it is inherently harder. The challenges are both theoretical (e.g., partial observability and lack of the Markov property) and practical (MARL algorithms often suffer from inferior stability and scalability).

In this work, we take a step towards amending this situation. We propose MA-Trace, a new on-policy actor-critic algorithm, which adheres to the centralized training and decentralized execution paradigm [10, 19, 25]. The key component of MA-Trace is the usage of importance sampling. This mechanism, based on V-Trace [8], provides off-policy correction for training data. As we demonstrate empirically, it allows distributing the computations efficiently in a multi-worker setup. Another advantage of MA-Trace is the fact that it is theoretically grounded. We provide a fixed-point theorem that guarantees convergence.

The on-policy algorithms directly optimize the objective; thus, they tend to be more stable and robust to hyperparameter choices than off-policy methods [1, 33, 35]. However, it is often impractical to train an on-policy algorithm in the distributed setting. When data collection is performed using many workers, the communication latency, asynchronicity, and other factors make the behavioral policies lag behind the target one. This results in a shift of the collected data towards off-policy distribution, which hurts the training quality. V-Trace reduces this shift by utilizing importance weights, thus permitting highly scalable training. Following that scheme, MA-Trace can be distributed to many workers to vastly reduce the wall-time of training with no negative impact on the results.

We evaluate MA-Trace on StarCraft Multi-Agent Challenge [27] – a standard benchmark for multi-agent algorithms. Our approach achieves competitive performance on all tasks and exceeds state-of-the-art results on some of them. Additionally, we provide a comprehensive set of ablations to quantify the influence of each component on the final results. We confirm that importance sampling is a key factor for MA-Trace's performance and show that our algorithm scales favorably with the number of actor workers. Additionally, we provide a few quite surprising findings, e.g. that an observation-based critic network performs better than a state-based.

For the description of key ideas and videos, visit our webpage: https://sites.google.com/view/ma-trace/main-page. The code used for our experiments is available at https://github.com/awarelab/seed_rl.

Our main contributions are the following:

(1) We introduce MA-Trace – a simple, scalable and effective multi-agent reinforcement learning algorithm with theoretical guarantees.
(2) We confirm that the training of MA-Trace can be easily distributed on multiple workers with nearly perfect speed-up and no negative impact on the quality.
(3) We provide extensive experimental validation of the MA-Trace algorithm in StarCraft Multi-Agent Challenge, including ablations with regard to importance sampling, centralization of learning, scaling and sharing of parameters.

2 RELATED WORK

For a general overview of multi-agent reinforcement learning (MARL) we refer to [5, 15]. Unsurprisingly, the development of MARL methods is closely coupled with the algorithmic progress in RL. A simple approach to multi-agent learning was proposed by Tan [34]: the IQL algorithm uses independent Q learners for each agent, with improvements proposed in [11, 17, 24].

MA-Trace adheres to the centralized training and decentralized execution (CTDE) paradigm. CTDE [9, 16, 23] is based on using the centralized information during training. During execution, the agents act using only their respective observations. Following this scheme, [9] introduces the RIAL and DIAL algorithms in the context of Q-learning. CTDE is particularly easy to implement with actor-critic algorithms; the centralized information is imputed only to the critic network (which is not used during the execution). COMA [10] is an example of such an algorithm; additionally, it uses a counterfactual baseline to deal with multi-agent credit assignment explicitly.

Another approach to take advantage of the multi-agent structure is the *value decomposition* method. VDN [32] propose a linear decomposition of the collective Q function into agent-local Q functions. Following this idea, [25] introduced QMIX, which learns a complex state-dependent decomposition by using monotonic mixing hypernetworks. Extensions of QMIX include MAVEN [21], COMIX [6], SMIX(λ) [39], and QTRAN [31] that can represent even general non-monotonic factorizations.

MA-Trace is based on V-Trace [8], a distributed single-agent algorithm. The idea of extending RL algorithms to the multi-agent setting has been successfully executed multiple times. Lowe et al. [19] propose a multi-agent actor-critic algorithm MADDPG, which is based on the DDPG algorithm [18]. Yu et al. [40] introduce also MASAC, extending SAC [14], and MATD3 building on top of TD3 [12]. Recently [41] showed that MAPPO, a multi-agent version of PPO [29], achieves surprisingly strong results in the most popular benchmarks, comparable with off-policy methods.

Espeholt et al. [8] propose the V-Trace algorithm to address the problem that in distributed (e.g. multi-node) training the policy used to generate experience is likely to lag behind the policy used for learning. Munos et al. [22] considered earlier a similar off-policy corrections for the target of the Q-function. These corrections are intended to focus on samples generated by behavioral policies close to the target one. Leaky V-Trace, a more general version of the V-Trace correction, was considered by Zahavy et al. [42]. Vinyals et al. [37] adapt V-Trace importance corrections to large action space to train grandmaster level StarCraft II agents. These corrections are refinements of the concept of importance sampling; see [33, Sections 5.5, 12.9] for a broader discussion.

Blending all these concepts, DOP [38] utilizes value decomposition and importance sampling to successfully train decentralized agents with policy gradients on off-policy samples. This is substantially different from our work since in MA-Trace we use importance weights to enable efficient multi-node training. DOP does not consider distributing the computations, the objective optimized by that algorithm requires providing on-policy samples, which is impossible to satisfy in a highly distributed setting.

3 BACKGROUND

Multi-agent reinforcement learning task is formalized by *decentralized partially observable Markov decision processes* (Dec-POMDP) [23]. A Dec-POMDP is defined as a tuple $(\mathcal{N}, \mathcal{S}, \mathcal{A}, P, r, \mathcal{Z}, O, \gamma, \rho_0)$. $\mathcal{N}$ is the set of agents $\{1, \ldots, n\}$, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions available to agents, P is the transition kernel, r is the reward function, $\mathcal{Z}$ is the space of collective observations, O is the set of observation functions $\{O_1, \ldots, O_n\}$, γ is the discount factor and ρ_0 is the initial state distribution. At state $s \in \mathcal{S}$, the agents select actions $a_i \sim \pi_i(\cdot|O_i(s))$, where π_i are their respective polices. Fix $a := (a_1, \ldots, a_n)$. The agents receive rewards according to the reward functions $r_i = r_i(s, a)$ and the system evolves to the next step generated by $P(s, a)$ (might be stochastic). In the so-called fully cooperative setting, assumed in this work, the rewards are equal, i.e. $r_1 = \ldots = r_n$.

The agents learn a joint policy

$$\pi(a|o) = \prod_{i=1}^{n} \pi_i(a_k|o_k) \tag{1}$$

with the aim to maximize the expected discounted return

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right].$$

The expected discounted return obtained by policy π starting from state $s \in \mathcal{S}$ is called the value function

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right], \tag{2}$$

4 MA-TRACE ALGORITHM

4.1 Overview of the algorithm

In this work, we introduce a multi-agent actor-critic algorithm based on V-trace: **MA-Trace**, see Algorithm 1. It follows the paradigm of centralized training, decentralized execution. Each agent compute its action taking its local observation as input. On the other hand, the critic network operates only during training, so it does not need to obey decentralization requirements. Furthermore, it can utilize any kind of additional information. We study two versions of MA-Trace: with the critic $V : \mathcal{S} \to \mathbb{R}$ taking as inputs full states, and with the critic $V : \mathcal{Z} \to \mathbb{R}$ taking as input the joint observation of all agents, denoted respectively as MA-Trace (full) and MA-Trace (obs). MA-Trace (full) requires collecting states s_t in line 6 of Algorithm 1 and using them as input to V_ϕ in lines 10 and 14.

The value function V^π corresponding to policy π can be, for example, obtained by repeated application of the Bellman operator. This requires on-policy data. The central innovation of V-Trace in the single-player setting and MA-Trace in the multi-player setting is to allow for slightly off-policy data by utilizing importance sampling. To this end, we use the V-Trace-inspired policy evaluation operator $\mathcal{R}$, defined as

$$\mathcal{R}V(s) := V(s) + \mathbb{E}_\mu \left[\sum_{t=0}^{+\infty} \gamma^t (c_0 \cdots c_{t-1}) \rho_t (r_t + \gamma V(s_{t+1}) - V(s_t)) | s_0 = s \right], \tag{3}$$

where $c_t = c(s_t, a_t), \rho_t := \rho(s_t, a_t)$ are importance sampling corrections and $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+, \rho : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ are measurable functions. In our algorithms we specialize to

$$c_t := \min\left(\overline{c}, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}\right), \quad \rho_t := \min\left(\overline{\rho}, \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}\right), \tag{4}$$

where μ is a policy that collected the data and $\overline{c}, \overline{\rho}$ are hyperparameters (usually set to 1.0). Intuitively speaking, c_t controls the speed of training and ρ_t balances the learned value function between V^π and V^μ. These parameters are further discussed in Corollary 3. The operator $\mathcal{R}$ leads to n-step Monte-Carlo target v_t given the state s_t

$$v_t := V(s_t) + \sum_{u=t}^{t+n+1} \gamma^{u-t} \left(\prod_{i=t}^{u-1} c_i\right) \rho_u \big(r_u + \gamma V(s_{u+1}) - V(s_u)\big). \tag{5}$$

It is a random variable; the clipping with the min function in (4) is instrumental to reducing its variance and thus making it applicable in learning. A key advantage of MA-Trace is a significant reduction in the wall-time due to distributed data collection (see line 6 in Algorithm 1). From the algorithmic standpoint, the major problem to address is that the policy used for collection $\pi_{\theta'}$ might be outdated due to communication overheads. This is successfully achieved with the importance correction mechanisms described above.

We use the communication model proposed in [7, Figure 1, Figure 3]. It consists of a single learner and actor workers. The actors are simple loops around the environment, generating observations (and rewards) transmitted to the learner. The learner makes inferences (and sends back actions); moreover, it handles trajectory accumulation and training.

Algorithm 1 MA-Trace

Require: d density of training
α learning rate

1: **for** k **in** $0, \ldots, n-1$ **do**
2: $\omega_k \leftarrow$ random actor parameters
3: $\phi \leftarrow$ random critic parameters
4: **for** *epoch* **in** $0, 1, 2, \ldots$ **do**
5: $\mathcal{D} \leftarrow \emptyset$
6: add trajectories $\{\tau_i\}$ sampled with $\pi_{\theta'}$ to $\mathcal{D}$
7: ▷ collected by multiple workers possibly remote.
8: **for** i **in** $0, \ldots, d-1$ **do**
9: sample $s_t \sim \mathcal{D}$
10: $\phi \leftarrow \phi - \alpha \nabla_\phi \left[\|v_t - V_\phi(s_t)\|_2^2\right]$ ▷ v_t is calculated according to (5)
11: **for** i **in** $0, \ldots, d-1$ **do**
12: sample $s_t \sim \mathcal{D}$
13: **for** k **in** $1, \ldots, n$ **do**
14: $g_k \leftarrow \rho_t \nabla_\omega \log(\pi_\omega(a_{t,k}|s_{t,k}))(r_{t,k} + \gamma V_\phi^k(s_{t+1}) - V_\phi^k(s_t))$
15: ▷ ρ_t is calculated according to (4)
16: $\omega_k \leftarrow \theta_k + \alpha g_k$

4.2 Theoretical analysis of MA-Trace

The operator, $\mathcal{R}$ enjoys the fixed point property. We present a proof of the following Theorem in Appendix A.

Theorem 1. *Let c, ρ be such that for any $s \in \mathcal{S}, a \in \mathcal{A}$*

$$\rho(s,a) - c(s,a)\, \mathbb{E}_{a' \sim \mu(\cdot|s')} \left[\rho(s', a')\right] \geq 0, \tag{6}$$

where s' is the state obtained from s after issuing action a. Assume also that $\mathbb{E}_\mu\, \rho_0 \geq \beta \in (0, 1]$. Then the operator $\mathcal{R}$ is a C_∞ contraction with a unique fixed point $V^{\tilde{\pi}}$ which is a value function of a policy $\tilde{\pi}$ given by

$$\tilde{\pi}(a|s) := \frac{\rho(s,a)\mu(a|s)}{\sum_{b \in \mathcal{A}} \rho(s,b)\mu(b|s)}. \tag{7}$$

The contraction constant is smaller than $1 - (1-\gamma)\beta < 1$.

Remark 2. *The theorem is an extended version of [8, Theorem 1]. First, we assume the vectorized statement, which is natural for the multi-agent setting. Second, the condition* (6) *admits more general importance sampling weights. We also fix a mathematical inaccuracy present in the original proof of [8, Theorem 1], see Remark 6.*

Now we can easily show that the result follows for the importance weights used in our work.

Corollary 3. *Let c_t, ρ_t be importance sampling weights* (4) *and $0 \leq \overline{c} \leq \overline{\rho}$. Assume also that $\mathbb{E}_\mu\, \rho_0 \geq \beta \in (0, 1]$. Then the operator $\mathcal{R}$ is a C_∞ contraction with a unique fixed point $V^{\tilde{\pi}}$ which is a value function of a policy $\tilde{\pi}$ given by*

$$\tilde{\pi}(a|s) := \frac{\min\left(\overline{\rho}\mu(a|s), \pi(a|s)\right)}{\sum_{b \in \mathcal{A}} \min\left(\overline{\rho}\mu(b|s), \pi(b|s)\right)}.$$

The contraction constant is smaller than $1 - (1-\gamma)\beta < 1$.

Proof. It is easy to check that $c(s,a) = \min\left(\overline{c}, \frac{\pi(a|s)}{\mu(a|s)}\right), \rho(s,a) = \min\left(\overline{\rho}, \frac{\pi(a|s)}{\mu(a|s)}\right)$ yield the importance sampling weights (4). Moreover, for any $s' \in \mathcal{S}$

$$\begin{aligned}
\mathbb{E}_{a' \sim \mu(\cdot|s')} \left[\rho(s', a')\right] &= \mathbb{E}_{a' \sim \mu(\cdot|s')} \min\left(\overline{\rho}, \frac{\pi(a'|s')}{\mu(a'|s')}\right) \\
&\leq \mathbb{E}_{a \sim \mu(\cdot|s')} \left(\frac{\pi(a'|s')}{\mu(a'|s')}\right) \\
&= 1
\end{aligned}$$

and therefore (6) holds whenever $0 \leq \overline{c} \leq \overline{\rho}$. Now the result follows by Theorem 1. □

Observe that when $\overline{\rho}$ is infinite, the fixed point $V^{\tilde{\pi}}$ corresponds to the target policy π. On the other hand, when $\overline{\rho}$ tends to 0, $V^{\tilde{\pi}}$ gets close to the value of the behavioral policy μ. In the general case, when $\overline{\rho}$ is finite but positive, the fixed point is the value function of a policy located somewhere between π and μ. However, $\tilde{\pi}$ does not depend on c_t – these weights affect the speed of convergence only [8].

Remark 4. *There is a theoretical difference between MA-Trace (full) and MA-Trace (obs), which perhaps is subtle in some cases. The Markov property is a key element required in the proof of Corollary 3. While it is by definition true for MA-Trace (state) it might fail for MA-Trace (obs) - if the concatenated observations do not provide a sufficient statistic of s_t.*

For the actor network we use policy gradient updates. Here we also need importance sampling to correct for using the off-policy behavioral policy μ. We recall the factorization (1); analogously we denote joint decentralized parameterized policy $\pi_\omega(a_1, \ldots, a_K|s) = \prod_{k=1}^{K} \pi_\omega(a_i|s_i)$. The policy gradient theorem suggests the ascent in the direction:

$$g := \mathbb{E}_{a_t \sim \pi_\omega} \left[\nabla_\omega \log \pi_\omega(a_t|s_t) A^{\pi_\omega}(s_t, a_t)) \right],$$

where $a_t = (a_{t,1}, \ldots, a_{t,K})$ and A^{π_ω} is some advantage estimator. For off-policy data collected with μ we have

$$g \approx \mathbb{E}_{a_t \sim \mu} \left[\rho_t \nabla_\omega \log \pi_\omega(a_t|s_t) A^{\pi_\omega}(s_t, a_t)) \right],$$

where the equality holds if $\bar{c} = +\infty$ in (4). This formula leads to the practical Monte-Carlo estimator used in line 14 of Algorithm 1:

$$\rho_t(\nabla_{\omega_i} \log(\pi_{\omega_i}(a_{t,i}|s_{t,i})))(r_t + \gamma V(s_{t+1}) - V(s_t)). \tag{8}$$

5 EXPERIMENTS

5.1 Environment

We evaluate MA-Trace on StarCraft Multi-Agent Challenge (SMAC) [28] (version 4.10), which is a standard benchmark for multi-agent algorithms, based on a popular real-time strategy game StarCraft II. It provides 14 micromanagement tasks of varying difficulty and structure.

The aim is to win a battle against a built-in AI by using your team of agents. In easier tasks, often rudimentary coordination is enough. However, harder tasks involve engaging a stronger enemy (e.g., having more units), which requires inventing smart techniques and tricks. Each unit has a limited line of sight, which makes the environment partially observable. We provide more details in Appendix F.

5.2 Main result

In Table 1 and Figure 1, we present the results of the main version of our algorithm – MA-Trace (obs), in which the critic uses stacked observations of agents as described in Appendix F. MA-Trace (obs) reaches competitive results and in some cases exceeds the current state-of-the-art. We compare with a selection of the state-of-the-art algorithms on SMAC following [41] and [26]. We also demonstarte scalability, which lets MA-Trace can reach good performance even using short training (wall time).

5.3 Training details

We use standard feed-forward networks for the actor and critic networks with two hidden layers of 64 neurons and ReLU activations. The critic network of MA-Trace (obs) takes stacked observations of agents as input, while MA-Trace (full) utilizes the full state provided by SMAC. DecMA-Trace have a critic using single-agent observations. See details in Appendix C.

For each reported version of MA-Trace, we have searched for the best hyperparameters to ensure a fair comparison. The values of all hyperparameters are listed in Appendix D.2.

To estimate the performance of MA-Trace we run training for 3 days or until convergence. We report the median win rate of 10 runs (with different random seeds) along with the interquartile range. Training curves for all the tasks can be found in Appendix G.

Task	MA-Trace (our)	MAPPO	IPPO	QMIX	COMA	IQL
2s3z	99 [99.5; 99.7]	100	100	95	43	75
3s5z	97 [96.6; 98.6]	100	100	88	1	10
1c3s5z	100 [99.8; 100]	100	100	96	31	21
5m_vs_6m	78 [74.3; 78.4]	89	87	75	1	49
10m_vs_11m	96 [86.2; 97.7]	97	93	95	7	34
27m_vs_30m	99 [99; 100]	94	69	39	0	0
3s5z_vs_3s6z	87 [81.6; 92.1]	84	83	83	0	0
MMM2	98 [98; 98.8]	90	87	87	0	0
2s_vs_1sc	99 [99; 99.6]	100	100	97	98	100
3s_vs_5z	0 [0; 0]	100	100	98	0	45
6h_vs_8z	85 [71; 88.8]	88	84	9	0	0
bane_vs_bane	100 [100; 100]	100	100	100	64	99
2c_vs_64zg	98 [98; 98.5]	100	98	92	0	7
corridor	91 [88.6; 96.1]	100	98	84	0	0

Table 1: Median win rate of MA-Trace (obs) compared with other algorithms. In *3s_vs_5z*, our agent discovers that keeping the opponents alive leads to higher rewards than killing them. This strategy, however, yields a low win rate. See Appendix F.1 for a detailed study.

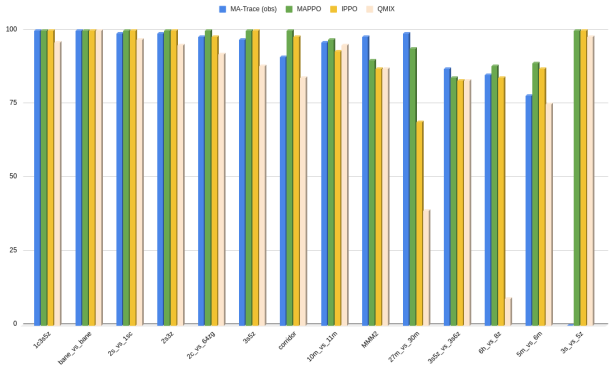


Figure 1: MA-Trace compared with state-of-the-arts algorithms on SMAC.

5.4 Ablations

Below we present a comprehensive list of ablations to evaluate the design choices of our algorithm. In each case, we present training curves for tasks, which best illustrate our claims. For the complete training results and more details, we refer to Appendix E.

Advantage of using importance sampling. Using the importance weights is the key algorithmic innovation of MA-Trace (and V-Trace), responsible for the strong performance we report. Indeed, already for 30 actor workers, using the weights is essential. Otherwise, the algorithm is unstable and suffers from poor asymptotic performance. See Figure 2 and Appendix E.2.

Training scaling. The importance sampling enables V-Trace to be truly scalable in multi-node setups. MA-Trace enjoys the same property. Importantly, we do not observe any degradation in the training performance when trained in the multi-node setup. See Figure 3 and Appendix E.3.

Input for the critic network. We found that MA-Trace (full) performs slightly worse than MA-Trace (obs). Usually the differences are small. However, in two harder tasks, *corridor* and *6h_vs_8z*,

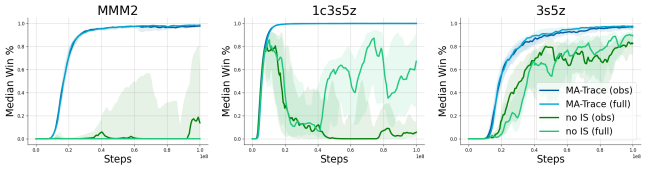


Figure 2: MA-Trace using 30 distributed workers with and without importance sampling (no IS).

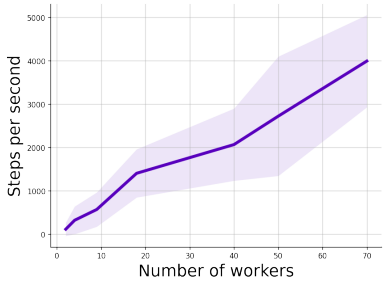


Figure 3: Speed of MA-Trace training with respect to the number of distributed workers, with standard deviation shaded. The speed is measured as the average number of steps processed per second.

MA-Trace (full) learns much slower and often fails. This is perhaps surprising, as the full state contains additional information (e.g., about invisible opponents). To deepen the analysis, we ran MA-Trace (obs+full), which uses both the observations and full state as the critic input. This improves the results, though they are still slightly inferior to MA-Trace (obs); see Figure 4 and Appendix G. A more detailed discussion of this topic can be found in Appendix E.1.

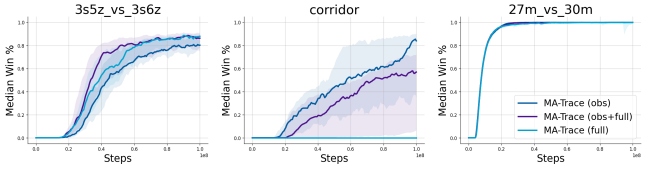


Figure 4: Comparison of using the full state MA-Trace (full) and aggregated agents' observations MA-Trace (obs) and both.

Centralized vs decentralized. As noted by [20], centralized training in some cases may suffer from higher variance. Therefore we compared MA-Trace with its decentralized version (i.e. having indpendent critics for each agent). The latter typically obtains weaker results and is less stable. See Figure 5 and details in Appendix B.

Sharing actors networks. We follow a common approach of sharing the policy network between agents. In some works, e.g., [26], to preserve individuality, the observations are enriched with agent ID. This might be beneficial if agents should be assigned different roles within the team. However, we find these benefits rather minor and opt for input provided by the environment (i.e., without ID). See Figure 7 and details in Appendix C.2.

One can also use separate networks for each agent. We check that MA-Trace works considerably worse in such a case. In rare

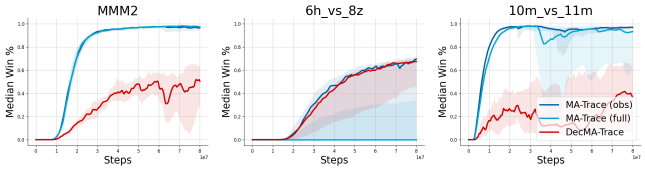


Figure 5: Performance of MA-Trace during centralized and decentralized training.

cases, using separate networks is advantageous, but only in the easiest tasks, e.g., *3s5z*. See Figure 6 and details in Appendix C.1.

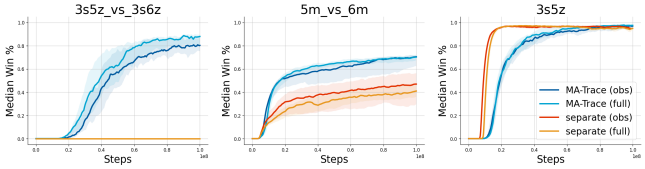


Figure 6: Performance of shared (standard MA-Trace) and separate agents' networks.

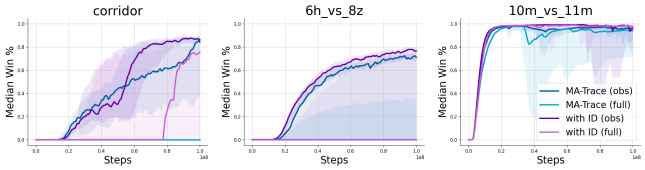


Figure 7: The impact of enriching observation with agent ID.

6 LIMITATIONS AND FURTHER WORK

We show that MA-Trace successfully solves SMAC tasks. Further benchmarking is needed to underpin its quality. This includes testing on more environments, both fully cooperative (like SMAC) and competitive. The latter might require further algorithmic developments.

The importance sampling weights successfully reduce distributional shifts arising in distributed training. An interesting question is whether they can also reduce the shifts introduced by the non-stationarity of the multi-agent environment.

MA-Trace exhibits lower sample efficiency than the other methods we used for comparisons. This, at least partially, can be explained by its on-policy nature. Adapting the importance correction to accommodate more off-policy data would be an important achievement.

7 CONCLUSIONS

In our work, we introduced MA-Trace, a new multi-agent reinforcement learning algorithm. We evaluated it on 14 scenarios constituting the StarCraft Multi-Agent Challenge and confirmed its strong performance. We also included ablations regarding importance sampling, centralization of learning, scaling, and sharing of parameters.

Thanks to the use of importance weights, MA-Trace is highly scalable. Furthermore, the convergence properties of our algorithm highlighted by Theorem 1 show that it has not only experimental but also mathematical grounding.

ACKNOWLEDGMENTS

We thank Konrad Staniszewski for discussions and experiments in the prequel of this project.

The work of Michał Zawalski was supported by the Polish National Science Center (NCN) grant UMO-2019/35/O/ST6/03464. The work of Henryk Michalewski was supported by the Polish National Science Center grant UMO-2018/29/B/ST6/02959. The work of Piotr Miłoś was supported by the NCN grant UMO-2017/26/E/ST6/00622. This research was supported by the PL-Grid Infrastructure. Some experiments were performed using the Entropy cluster funded by NVIDIA, Intel, the Polish National Science Center grant UMO-2017/26/E/ST6/00622 and ERC Starting Grant TOTAL. Our experiments were managed using https://neptune.ai. We would like to thank the Neptune team for providing access to the team version and technical support. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH, PCSS) for providing computer facilities and support within computational grant no. PLG/2021/014561.

REFERENCES

[1] Josh Achiam. 2018. Spinning Up in Deep RL. https://spinningup.openai.com.

[2] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. 2020. Learning dexterous in-hand manipulation. *Int. J. Robotics Res.* 39, 1 (2020). https://doi.org/10.1177/0278364919887447

[3] Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 449–458. http://proceedings.mlr.press/v70/bellemare17a.html

[4] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *CoRR* abs/1912.06680 (2019). arXiv:1912.06680 http://arxiv.org/abs/1912.06680

[5] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[6] Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2020. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709* (2020).

[7] Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski. 2020. SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rkgvXlrKwH

[8] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 1406–1415. http://proceedings.mlr.press/v80/espeholt18a.html

[9] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 2137–2145. https://proceedings.neurips.cc/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html

[10] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 2974–2982. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17193

[11] Jakob N. Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 1146–1155. http://proceedings.mlr.press/v70/foerster17b.html

[12] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 1582–1591. http://proceedings.mlr.press/v80/fujimoto18a.html

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 1856–1865. http://proceedings.mlr.press/v80/haarnoja18b.html

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, 1861–1870.

[15] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. 2020. A Very Condensed Survey and Critique of Multiagent Deep Reinforcement Learning. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, Amal El Fallah Seghrouchni, Gita Sukthankar, Bo An, and Neil Yorke-Smith (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 2146–2148. https://dl.acm.org/doi/abs/10.5555/3398761.3399105

[16] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[17] Martin Lauer and Martin A. Riedmiller. 2000. An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, Pat Langley (Ed.). Morgan Kaufmann, 535–542.

[18] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1509.02971

[19] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6379–6390. https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html

[20] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. 2021. Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning. *CoRR* arXiv/2102.04402 (2021). arXiv:2102.04402 https://arxiv.org/abs/2102.04402

[21] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 7611–7622. https://proceedings.neurips.cc/paper/2019/hash/f816dc0acface7498e10496222e9db10-Abstract.html

[22] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. 2016. Safe and Efficient Off-Policy Reinforcement Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D.

Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 1046–1054. https://proceedings.neurips.cc/paper/2016/hash/c3992e9a68c5ae12bd18488bc579b30d-Abstract.html

[23] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer. https://doi.org/10.1007/978-3-319-28929-8

[24] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*. PMLR, 2681–2690.

[25] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research)*, Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 4292–4301. http://proceedings.mlr.press/v80/rashid18a.html

[26] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *CoRR* abs/2003.08839 (2020). arXiv:2003.08839 https://arxiv.org/abs/2003.08839

[27] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 2186–2188. http://dl.acm.org/citation.cfm?id=3332052

[28] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. *CoRR* abs/1902.04043 (2019).

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

[30] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George 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. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (01 2016), 484–489. https://doi.org/10.1038/nature16961

[31] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. PMLR, 5887–5896. http://proceedings.mlr.press/v97/son19a.html

[32] 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. 2017. Value-Decomposition Networks For Cooperative Multi-Agent Learning. arXiv:cs.AI/1706.05296

[33] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[34] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, Paul E. Utgoff (Ed.). Morgan Kaufmann, 330–337. https://doi.org/10.1016/b978-1-55860-307-3.50049-6

[35] John N. Tsitsiklis and Benjamin Van Roy. 1997. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Autom. Control.* 42, 5 (1997), 674–690. https://doi.org/10.1109/9.580874

[36] Oriol Vinyals, I. Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, J. Chung, D. Choi, Richard Powell, Timo Ewalds, P. Georgiev, Junhyuk Oh, Dan Horgan, M. Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, J. Agapiou, Max Jaderberg, A. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, D. Budden, Yury Sulsky, James Molloy, T. Paine, Caglar Gulcehre, Ziyu Wang, T. Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* (2019), 1–5.

[37] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[38] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2020. Off-Policy Multi-Agent Decomposed Policy Gradients. *CoRR* abs/2007.12322 (2020). arXiv:2007.12322 https://arxiv.org/abs/2007.12322

[39] Chao Wen, Xinghu Yao, Yuhui Wang, and Xiaoyang Tan. 2020. SMIX(λ): Enhancing Centralized Value Functions for Cooperative Multi-Agent Reinforcement Learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 7301–7308. https://aaai.org/ojs/index.php/AAAI/article/view/6223

[40] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2020. Benchmarking Multi-agent Deep Reinforcement Learning Algorithms. Workshop in Conference on Neural Information Processing Systems 2020. https://www.researchgate.net/publication/349943157_Benchmarking_Multi-agent_Deep_Reinforcement_Learning_Algorithms

[41] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. 2021. The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games. *CoRR* abs/2103.01955 (2021). arXiv:2103.01955 https://arxiv.org/abs/2103.01955

[42] Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. 2020. A Self-Tuning Actor-Critic Algorithm. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/f02208a057804ee16ac72ff4d3cec53b-Abstract.html

A THE PROOF OF THE FIXED POINT THEOREM

Let C_∞^K be the space of functions $\mathcal{S} \mapsto \mathbb{R}^K$ endowed with the infinity norm $\|f\|_{C_\infty^K} = \sup_{s\in\mathcal{S}} |f(s)|_\infty$. C_∞^K is Banach if $\mathcal{S}$ is finite or we additionally assume that functions are continuos (this encompasses the case when $\mathcal{S}$ is discrete).

For an easy reference we recall the statement of Theorem 1. Let $c : \mathcal{S}\times\mathcal{A} \to \mathbb{R}_+$, $\rho : \mathcal{S}\times\mathcal{A} \to \mathbb{R}_+$ be measurable functions and set $c_t = c(s_t, a_t)$, $\rho_t = \rho(s_t, a_t)$.

Define the operator $\mathcal{R} : C_\infty^K \mapsto C_\infty^K$ by

$$\mathcal{R}V(s) := V(s) + \\ + \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t (c_0 \cdots c_{t-1})\rho_t (r_t + \gamma V(s_{t+1}) - V(s_t))|s_0 = s\right]. \tag{9}$$

Theorem 5. *Let c, ρ be such that for any $s \in \mathcal{S}, a \in \mathcal{A}$*

$$\alpha(s,a) := \rho(s,a) - c(s,a)\,\mathbb{E}_{a'\sim\mu(\cdot|s')}\left[\rho(s',a')\right] \geq 0, \tag{10}$$

where s' is the state obtained from s after issuing action a. Assume also that $\mathbb{E}_\mu \rho_0 \geq \beta \in (0,1]$. Then the operator $\mathcal{R}$ is C_∞^K contraction with a unique fixed point $V^{\tilde\pi}$ which is a value function of a policy $\tilde\pi$ given by

$$\tilde\pi(a|s) := \frac{\rho(s,a)\mu(a|s)}{\sum_{b\in\mathcal{A}} \rho(s,b)\mu(b|s)}. \tag{11}$$

The contraction constant η is smaller than $1 - (1-\gamma)\beta < 1$.

Remark 6. *We note that the proof [8, Theorem 1] has some glitches. A careful examination reveals that the analysis of the first displayed equation on page 12 in [8] is not correct. We fix it by making introducing filtration and analyzing the measurability of explicitly and argue that such an approach makes the proof more clear*

Proof. Let $\tilde c_t := \prod_{u=0}^t c_u$ with the convention $\tilde c_t = 1$ for $t \leq 0$. We use the same convention for ρ_s. It is convenient to express $\mathcal{R}$ in the following form

$$\mathcal{R}V(s) = (1 - \mathbb{E}_\mu \rho_0)V(s) + \\ + \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-1}\left(\rho_t r_t + \gamma[\rho_t - c_t\rho_{t+1}]V(s_{t+1})\right)\right].$$

Fix $V_1, V_2 \in C_\infty^K$ and put $C_\infty^K \ni \Delta V := V_1 - V_2$. For $s \in \mathcal{S}$ we write

$$\mathcal{R}V_1(s) - \mathcal{R}V_2(s) = (1 - \mathbb{E}_\mu \rho_0)\Delta V(s) + \\ + \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^{t+1} \tilde c_{t-1}[\rho_t - c_t\rho_{t+1}]\Delta V(s_{t+1})\right].$$

We define filtration $\{\mathcal{F}_t\}_{t=0}^{+\infty}$ by

$$\mathcal{F}_t := \sigma((s_0, a_0, \ldots, a_{t-1}, s_t)).$$

We rewrite in terms of conditional expectation

$$a := \mathbb{E}_\mu\left[\tilde c_{t-1}[\rho_t - c_t\rho_{t+1}]\Delta V(s_{t+1})\right] \\ = \mathbb{E}_\mu\left(\mathbb{E}_\mu\left[\tilde c_{t-1}[\rho_t - c_t\rho_{t+1}]\Delta V(s_{t+1})|\mathcal{F}_{t+1}\right]\right).$$

Clearly, all terms except for ρ_{t+1} are $\mathcal{F}_{t+1}$ measurable, thus

$$a = \mathbb{E}_\mu\left[\tilde c_{t-1}[\rho_t - c_t\,\mathbb{E}_\mu\left(\rho_{t+1}|\mathcal{F}_{t+1}\right)]\Delta V(s_{t+1})\right].$$

Recall (10) and observe that $\rho_t - c_t\,\mathbb{E}_\mu\left(\rho_{t+1}|\mathcal{F}_{t+1}\right) = \alpha(s_t, a_t) =: \alpha_{t+1}$. Let us also put by convention $\alpha_0 = 1 - \mathbb{E}_\mu \rho_0$. Thus shifting indices we get

$$\mathcal{R}V_1(s) - \mathcal{R}V_2(s) = \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-2}\alpha_t \Delta V(s_t)\right].$$

By our assumptions we have $\tilde c_t, \alpha_t \geq 0$ and thus we get

$$|\mathcal{R}V_1(s) - \mathcal{R}V_2(s)| \leq \|V_1 - V_2\|_{C_\infty^K}\,\mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-2}\alpha_t\right].$$

We are left with rather straightforward calculations (assume by convention that $\rho_{-1} = 1$).

$$\begin{aligned}
\mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-2}\alpha_t\right] &= \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-2}\rho_{t-1}\right] - \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-1}\,\mathbb{E}_\mu\left(\rho_t|\mathcal{F}_t\right)\right] \\
&= \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-2}\rho_{t-1}\right] - \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-1}\rho_t\right] \\
&= 1 + \gamma\,\mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-1}\rho_t\right] - \mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-1}\rho_t\right] \\
&= 1 + (\gamma - 1)\,\mathbb{E}_\mu\left[\sum_{t=0}^{+\infty} \gamma^t \tilde c_{t-1}\rho_t\right] \leq 1 + (\gamma-1)\beta.
\end{aligned}$$

The last inequality follows by dropping all summands except $t = 0$.

Now, we are to determine the unique fixed point. Recall (11), we calculate

$$\begin{aligned}
&\mathbb{E}_\mu\left[\rho_t\left(r_t + \gamma V^{\tilde\pi}(s_{t+1}) - V^{\tilde\pi}(s_t)\right)|s_t\right] \\
&= c\,\mathbb{E}_{\tilde\pi}\left[\rho_t\left(r_t + \gamma V^{\tilde\pi}(s_{t+1}) - V^{\tilde\pi}(s_t)\right)|s_t\right] \\
&= 0,
\end{aligned}$$

where $c = \sum_{b\in\mathcal{A}} \rho(s,b)\mu(b|s)$ is the normalizing constant and the second equality hold by the Bellman equation. □

B CENTRALIZED TRAINING AND DECENTRALIZED EXECUTION

In a multi-agent problem, the agents take actions based on their local observations s_i, according to their policies π_i. **Decentralized training** is a simple approach of learning π_i, using any (single-agent) RL algorithm separately for each agent. To better utilize the structure of a multi-agent problem, a paradigm of **centralized training and decentralized execution** was introduced. The key idea is to centralize the learning process of all the agents – use shared knowledge provided by observations and any other information available during training to more effectively optimize the decentralized policies. It is now considered a leading paradigm and is usually chosen off-the-shelf.

Recently the authors of [20] suggested that this should be reinvestigated. Sharing the knowledge in centralized training is indeed beneficial, but the paradigm of independent training also has advantages. The multi-agent problems lack the strong theoretical guarantees associated with the standard case. For example, if we train all the agents simultaneously, the environment changes during training from every agent's perspective. Such non-stationarity can destabilize standard algorithms. According to [20], decentralized learning partially mitigates this issue – for every agent, a separate value network is trained; thus it averages much of the stochasticity in the environment, producing more stable estimates. On the other hand, because of the limited information, these values are less accurate. Therefore choosing between centralized and decentralized training is a tradeoff.

To address these issues, we compared MA-Trace with its decentralized version, DecMA-Trace. However, in the StarCraft Multi-Agent Challenge, the decentralized version learns much slower and fails to reach good performance; see Figure 8.

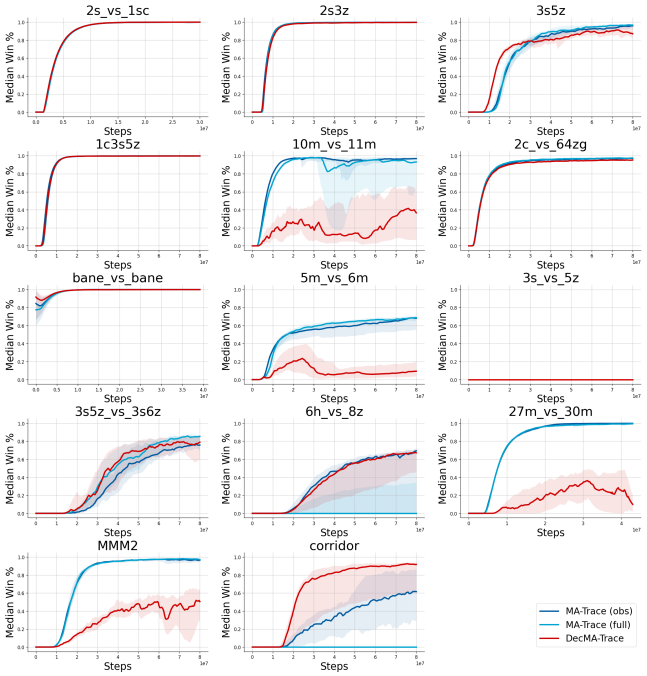


Figure 8: Comparison of MA-Trace and DecMA-Trace on the StarCraft Multi-Agent Challenge tasks.

C NETWORKS' ARCHITECTURES

In our experiments, we use feed-forward networks with two hidden layers of 64 neurons and ReLU activations (without normalization). This is smaller than the networks used in benchmark [40] and [26], but we found this sufficient to obtain good results. In this work, we focus on algorithmic aspects rather than tuning architectures. However, for completeness, we include a discussion of some most popular design choices.

C.1 Sharing parameters

Our default (and most effective) scheme uses a single shared network for the actors and a separate one for the critic. We experimented with sharing feature extractors between the agents and the critic, which performed worse. We also checked the performance when training separate networks for all the components. Interestingly, with this approach, the learning is much faster on the easy tasks, such as *3s5z* or *2c_vs_64zg*, but completely fails on the hardest ones; see Figure 9.

C.2 ID experiments

Sharing the agents' networks, as described in Section C.1, can lead to poor results if agents are not homogenous (for example, the behavior of shooting units should differ from melee units). This might be circumvented by adding units' characteristics (which is a default in SMAC) or enriching the observations with one-hot encoded ID. In Figure 10 we present the results of experiments in which we use these two mechanisms. We observe some improvements, which are, however, relatively minor.

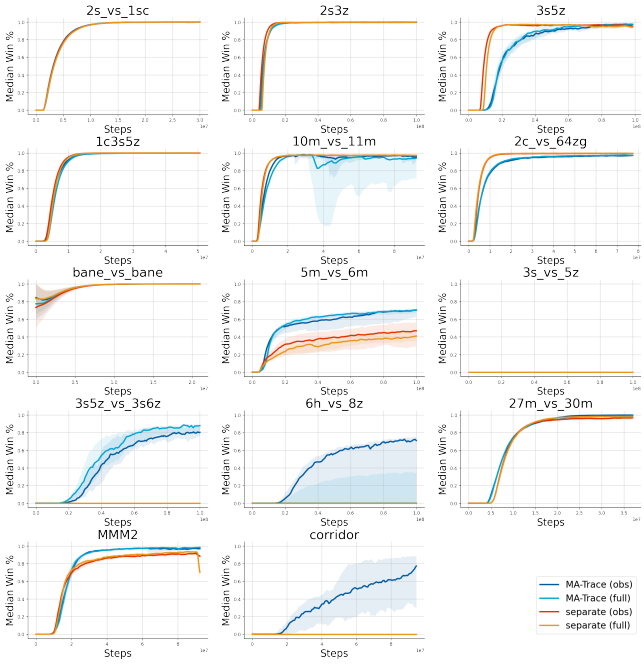


Figure 9: Comparison of standard MA-Trace (with a shared actor network) and its ablation with separate networks.

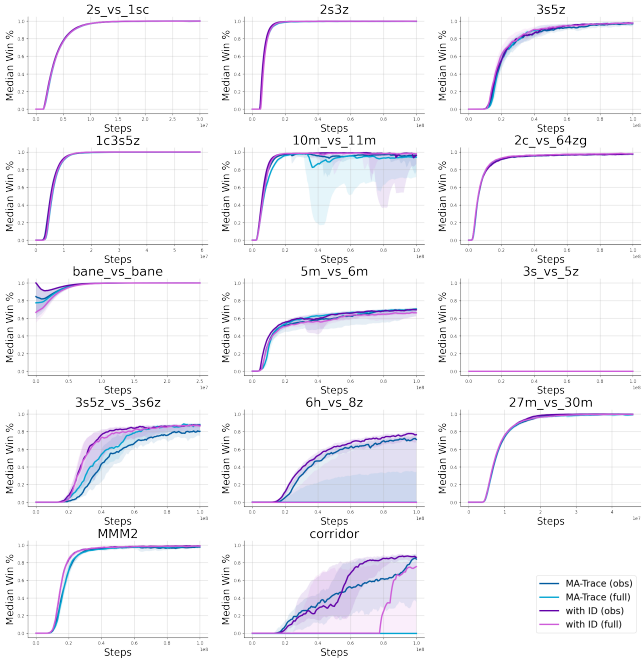


Figure 10: Comparison of MA-Trace conditioned on observation with and without (default) agents' IDs.

C.3 Agents with memory

The units in StarCraft Multi-agent Challenge have limited sight range, thus the environment is partially observable. A common approach to mitigate such an issue, is to give the agents memory, e.g. by using recurrent networks or frame stacking. The authors of [26] show that using recurrent policy is required to achieve good results in the hardest tasks. However, our experiments show that memoryless policies can be successfully trained to solve all the tasks (possibly except *3s_vs_5z*, see Section F.1).

To implement a simple memory it is enough to pass a few previous observations concatenated ("framestack"). We experimented with both observation-based and state-based critics. Figures 11 and 12 show the progress of training memory-equipped agents. One can observe hardly any difference in most tasks. On the easy task *2c_vs_64zg*, using memory leads to faster training. What is particularly interesting, on the *corridor*, task the stacked versions learn much faster. This may be related to the strategy learned by MA-Trace, in which at some point a group of units has to hide and wait (see Section F.2 for detailed description). Execution of this strategy is probably easier when using at least short memory. However, on some other tasks, such as *5m_vs_6m*, training with memory is much less stable and effective.

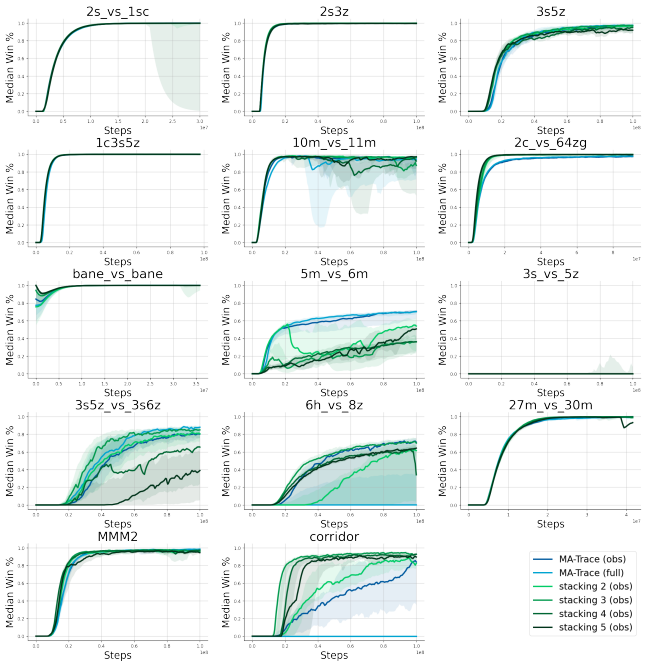


Figure 11: Comparison of MA-Trace with growing number of stacked frames. The critic networks in ablations are conditioned on aggregated observations.

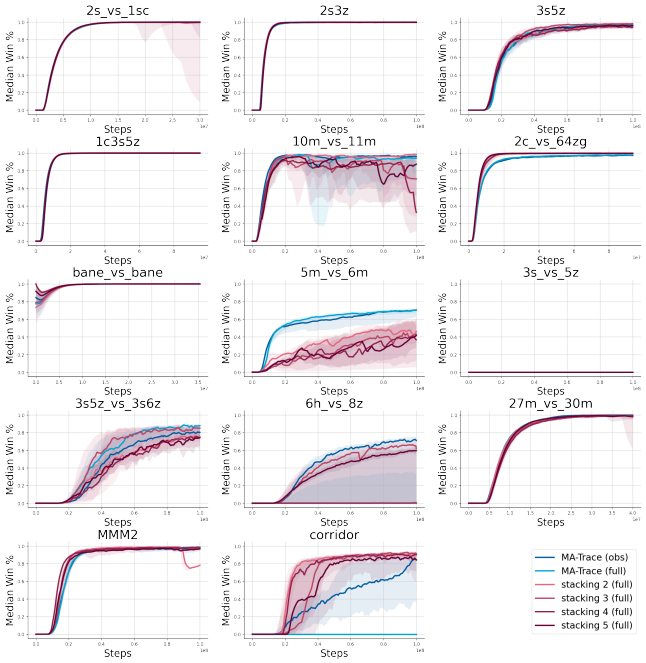


Figure 12: Comparison of MA-Trace with growing number of stacked frames. The critic networks in ablations are conditioned on environment state.

D TRAINING DETAILS

D.1 SEED RL

We base our code on SEED RL [7], which is an open-source framework for distributed learning, licensed under the Apache License, Version 2.0. This library provides an implementation of V-Trace for single-agent environments.

D.2 Hyperparameters

In our experiments the parameters crucial to the final performance were: *learning rate* and *entropy cost*. For other hyperparameters, we use the default values provided by SEED. The most relevant, common for all the experiments, are listed in Table 2.

hyperparameter	value
batch size	32
optimizer	Adam
gamma	0.99
ρ	1.0
c	1.0
λ	1.0
initial entropy cost	1.0
target entropy	10^{-5}

Table 2: Default parameters for our experiments.

We searched for the best learning rate in the set $[3.5 \cdot 10^{-3}, 2.5 \cdot 10^{-3}, 1.5 \cdot 10^{-3}, 10^{-3}, 7 \cdot 10^{-4}, 5 \cdot 10^{-4}, 3.5 \cdot 10^{-4}]$. We experimented with adding learning rate schedules with warmup and decay, though no such scheme appeared to be beneficial.

Another hyperparameter that has a strong influence on the final results is *entropy cost*. By default, SEED sets it to a constant value of $2.5 \cdot 10^{-4}$ and allows to use annealing. We found that aggressive exploration at the beginning is crucial to reach good results. We annealed *entropy cost* from 1 towards 10^{-5}. The speed of adjustment was tuned in the set $[10, 5, 2.5, 1, 0.5]$.

In Table 3, we show the best parameters for our experiments, found by the above grid-search. Unlike in some other works, we found no advantage of using gradient clipping; thus, we leave the gradients not clipped.

hyperparameter	MA-Trace (obs)	MA-Trace (full)
learning rate	10^{-3}	$7 \cdot 10^{-4}$
entropy adjustment	10	10

Table 3: Specific parameters for our experiments.

D.3 Infrastructure used

The typical configuration of a computational node used in our experiments was: the Intel Xeon E5-2697 2.60GHz processor with 128GB memory. On a single node, we ran one experiment with 30 workers. A typical experiment was run for about 20h. For the final evaluation, we extended the training to 3 days, which is usually equivalent to about $3 \cdot 10^8$ environment steps. We did not use GPUs; we found that with the relatively small size of the network it offers only slight wall-time improvement while generating substantial additional costs.

E ABLATIONS

E.1 Critic comparison

During centralized training, the critic network in MA-Trace algorithm uses any information available. A natural choice is to aggregate the observations available to the agents, and we denote this version as MA-Trace (obs). This might not be a sufficient statistic of the (Markov) state of the environment. SMAC provides additional access to such a state description, which we use in MA-Trace (full). Intuitively, using complete information should be advantageous.

As shown in Figure 16, on most tasks, both the versions do not differ much. On *3s5z_vs_3s6z* there is a small advantage on the full-state side. However, MA-Trace (full) shows little progress on *6h_vs_8z* and *corridor*, as opposed to MA-Trace (obs). Therefore we consider MA-Trace (obs) to be our default version.

Such behavior is a bit counter-intuitive. We speculate that some information available in the agents' observation is not easily accessible (computable) for the full state. To verify this, we compared the two versions with another, MA-Trace (obs+full), which uses both the aggregated observations and the full state. As we can see in Figure 16, it trains similarly to MA-Trace (obs), without significant advantage on any task. Moreover, on the hardest tasks, the training progresses a bit slower. We leave a precise explanation of this behavior as future work.

E.2 Importance sampling ablation

We claim that the strong performance of MA-Trace is due to using importance weights. To verify this statement, we compared MA-Trace with its ablation without importance weights. The training curves for all the tasks are shown in Figure 13. Without the corrections, the training is unstable and reaches good performance only on the easiest tasks. The difference gets even larger when using more compute workers. One notable exception is the *corridor* task, in which the ablated version trains faster than MA-Trace, though eventually both reach similar results.

E.3 Scaling experiments

MA-Trace utilizes importance sampling to reduce the shifts arising naturally in distributed training (when behavioral policies lag behind the target policy). This allows our algorithm to be highly scalable. Our experiments confirm that MA-Trace scales almost linearly on at least 70 workers.

As shown in Section E.2, the importance weights indeed enable stable training, even when distributed to many workers. What is equally important, our experiments show that scaling does not affect the learned policies significantly.

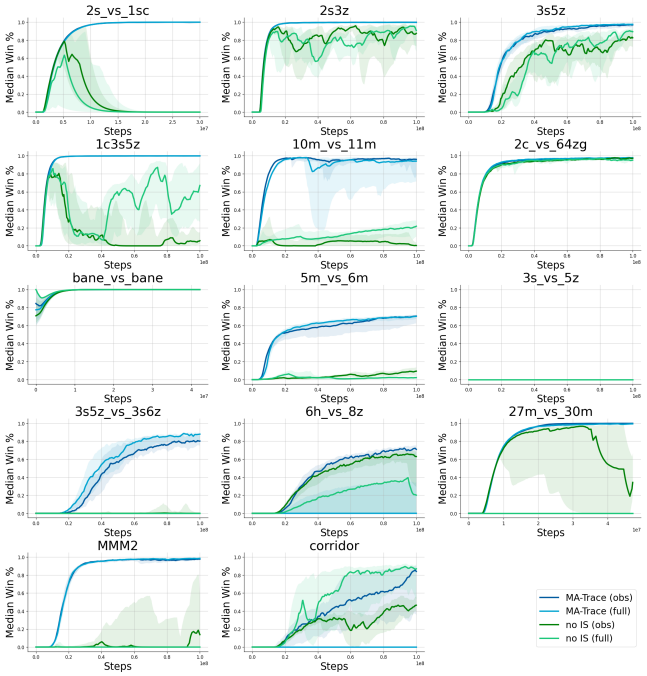


Figure 13: Comparison of MA-Trace with and without importance sampling.

F STARCRAFT MULTI-AGENT CHALLENGE

To evaluate MA-Trace, we use StarCraft Multi-Agent Challenge (SMAC) [28]. It is a standard benchmark for multi-agent algorithms, used e.g. by [10, 26, 40] and many others. The challenge is based on a popular real-time strategy game StarCraft II and consists of 14 micromanagement tasks. Each task is a small arena in which two teams, controlled by the player by the built-in AI, fight against each other. The goal in every task is to defeat (kill) all the enemy units.

Units belong to one of the three races: Protoss, Terran or Zerg. Additionally, they are divided on a number of classes with unique characteristic, such as speed, shooting range, fire power etc. In each turn they can move or attack an enemy in their shooting range. A unit is considered defeated if its health drops to 0. A defeated unit remain inactive.

SMAC provides a variety of different tasks. In the easier tasks, the opponents control the same forces. Therefore to win such a game, it is enough to coordinate slightly better than the built-in AI. In the harder tasks, however, the computer starts with a stronger squad. This can be a minor advantage, such as in the task *10 marines vs 11 marines*, or quite a big difference. In particularly hard scenarios, such as *corridor*, it is unreasonable for the player to engage in an open fight, so it is essential to develop a long-term strategy to obtain a positional advantage.

For the hardest tasks, the authors of the challenge consider the so-called microtricks sufficient to win consistently. However, it appears that MA-Trace in some cases develop unexpected strategies, see e.g. Section F.2. What is particularly interesting, our algorithm manages to learn some techniques associated with professional

human players, such as focusing fire, withdrawing low-health units, hit-and-run, sacrificing a unit to deceive the opponent and others.

Units in the game have limited sight range, which makes the environment partially observable. The observations received by individual agents contain information about all the visible units (including themselves) – their health, energy, position, class, and other relevant features. All the units beyond the sight range are marked as dead (i.e., defeated and invisible units are not distinguished). It is possible that the aggregated observations do not provide full information, for there can be enemy units hidden beyond the sight range of any ally. Therefore, to facilitate centralized training, SMAC provides additional access to the full state of the environment.

Learning from binary reward (win/lose) is prohibitively hard in most tasks. Therefore SMAC provides dense rewards to enable training. The team receives additional points for damage dealt and defeating a unit. This scheme is arguably natural and leads to successful training. However, in some cases, it might reinforce undesired behaviors, see, e.g., Section F.1.

F.1 *3s_vs_5z* task

MA-Trace masters all the tasks except *3s_vs_5z*, see Table 1. In that scenario, we control 3 Stalkers fighting against 5 Zealots. Stalkers can attack the enemy from a distance; however, they are no match for Zealots in close combat. A strategy to gain an advantage is to shoot the enemies while they are away and flee when they get close.

All the units in this task are Protoss, so they all have protective shields. The shields absorb some amount of damage, until they are down. However, as opposed to regular health, the shields regenerate slowly with time. As dealing damage yields rewards, it might be beneficial to keep enemy alive infinitely. Apparently, MA-Trace learns to this strategy.

Figure 14 shows an example of the winning rate and episode rewards. As we can see, after a short training, our algorithm wins almost every time. Further, it discovers that the reward scheme can be exploited – at some point, the mean return increases fast, while the actual win rate decreases and becomes unstable.

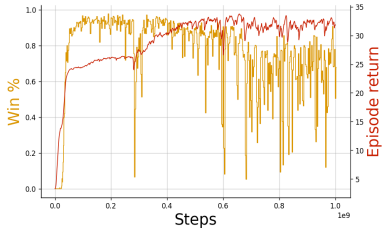


Figure 14: The winning rate and episode rewards in a training for the task *3s_vs_5z*.

F.2 *Corridor* task

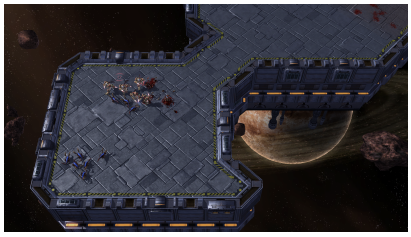
Another *super-hard* scenario (according to [26]) is *corridor*. In this task, we control a team of 6 Zealots against 24 enemy Zerglings. Though Zealots are far more powerful in combat, they are outnumbered; thus, the open fight is clearly an unreasonable strategy. However, the fighting arena contains a narrow passage. The authors of SMAC suggest that a winning strategy is to gather the forces in that passage, where the number of enemy units should lose its importance (possibly inspired by the Battle of Thermopylae).

However, MA-Trace develops an alternative interesting strategy. Firstly, our forces split into two groups. One (strong) hides in the corner, where it easily defeats a few enemies, while the other (one or two units) attracts majority of the enemies to the other side and sacrifice itself. After defeating the second group, the enemies pass to the far side of the arena and wait, unaware of the hidden group. Then the strong group attack them from behind and defeat the Zerglings one by one.

The strategy is outlined in Figure 15.

(a) Split into two groups.

(b) Hide the stronger, sacrifice the weaker.



(c) Attack from behind.

Figure 15: The consecutive parts of strategy executed by MA-Trace on the *corridor* task. The yellow soldiers are our units, while the blue creatures are enemy units.

G FULL TRAINING DATA

In this section, we provide the main practical results of our work – complete training data for all standard versions of MA-Trace on all the SMAC tasks. They were trained for three days or until convergence. We report the median win rates and interquartile ranges.

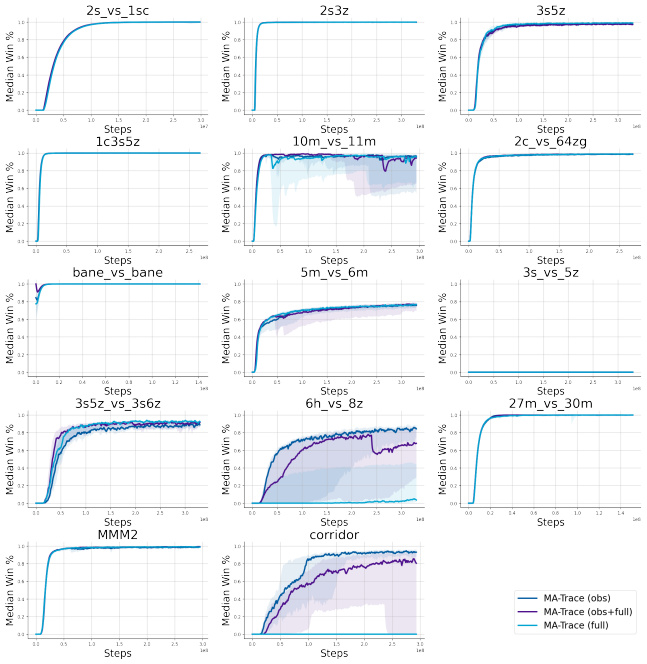


Figure 16: Training curves of main MA-Trace versions on all the tasks available in StarCraft Multi-Agent Challenge.