Towards a Standardised Performance Evaluation Protocol for Cooperative MARL

Rihab Gorsane[1*] **Omayma Mahjoub**[12*†] **Ruan de Kock**[1*] **Roland Dubb**[13†]

Siddarth Singh[1] **Arnu Pretorius**[1]

[1]InstaDeep
[2]National School of Computer Science, Tunisia
[3]University of Cape Town, South Africa

Abstract

Multi-agent reinforcement learning (MARL) has emerged as a useful approach to solving decentralised decision-making problems at scale. Research in the field has been growing steadily with many breakthrough algorithms proposed in recent years. In this work, we take a closer look at this rapid development with a focus on evaluation methodologies employed across a large body of research in cooperative MARL. By conducting a detailed meta-analysis of prior work, spanning 75 papers accepted for publication from 2016 to 2022, we bring to light worrying trends that put into question the true rate of progress. We further consider these trends in a wider context and take inspiration from single-agent RL literature on similar issues with recommendations that remain applicable to MARL. Combining these recommendations, with novel insights from our analysis, we propose a standardised performance evaluation protocol for cooperative MARL. We argue that such a standard protocol, if widely adopted, would greatly improve the validity and credibility of future research, make replication and reproducibility easier, as well as improve the ability of the field to accurately gauge the rate of progress over time by being able to make sound comparisons across different works. Finally, we release our meta-analysis data publicly on our project website for future research on evaluation [3] accompanied by our open-source evaluation tools repository[4].

1 Introduction

Empirical evaluation methods in single-agent reinforcement learning (RL)[5] have been closely scrutinised in recent years (Islam et al., 2017; Machado et al., 2017; Henderson, 2018; Zhang et al., 2018; Henderson et al., 2018; Colas et al., 2018a, 2019; Chan et al., 2020; Jordan et al., 2020; Engstrom et al., 2020; Agarwal et al., 2022). In this context, the impact of a lack of rigour and methodological standards has already been observed. Fortunately, just as these issues have arisen and been identified, they have also been accompanied by suggested solutions and recommendations from these works.

*Equal contribution. Corresponding author: r.gorsane@instadeep.com
†Work done during an internship at InstaDeep.
[3]https://sites.google.com/view/marl-standard-protocol
[4]https://github.com/instadeepai/marl-eval
[5]In this paper, we use the term "RL" to exclusively refer to *single-agent* RL, as opposed to RL as a field of study, of which MARL is a subfield.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Multi-agent reinforcement learning (MARL) extends RL with capabilities to solve large-scale decentralised decision-making tasks where many agents are expected to coordinate to achieve a shared objective (Foerster, 2018; Oroojlooyjadid and Hajinezhad, 2019; Yang and Wang, 2020). In this cooperative setting, where there is a common goal and rewards are shared between agents, sensible evaluation methodology from the single-agent case often directly translates to the multi-agent case. However, despite MARL being far less developed and entrenched than RL, arguably making adoption of principled evaluation methods easier, the field remains affected by many of the same issues as found in RL. Implementation variance, inconsistent baselines, and insufficient statistical rigour still affect the quality of reported results. Although work specifically addressing these issues in MARL have been rare, there have been recent publications that implicitly observe some of the aforementioned issues and attempt to address their symptoms but arguably not their root cause (Yu et al., 2021a; Papoudakis et al., 2021; Hu et al., 2022). These works usually attempt to perform new summaries of performance or look into the code-level optimisations used in the literature to provide insight into the current state of research.

In this paper, we argue that to facilitate long-term progress in future research, MARL could benefit from the literature found in RL on evaluation. The highlighted issues and recommendations from RL may serve as a guide towards diagnosing similar issues in MARL, as well as providing scaffolding around which a standardised protocol for evaluation could be developed. In this spirit, we survey key works in RL over the previous decade, each relating to a particular aspect of evaluation, and use these insights to inform potential standards for evaluation in MARL.

For each aspect of evaluation considered, we provide a detailed assessment of the corresponding situation in MARL through a meta-analysis of prior work. In more detail, our meta-analysis involved manually annotating MARL evaluation methodologies found in research papers published between 2016 to 2022 from various conferences including NeurIPS, ICML, AAMAS and ICLR, with a focus on *deep* cooperative MARL (see Figure 1). In total, we collected data from 75 cooperative MARL papers accepted for publication. Although we do not claim our dataset to comprise the entire field of modern deep MARL, to the best of our knowledge, our data includes all popular and recent deep MARL algorithms and methodologies from seminal papers. We believe this dataset is the first of its kind and we have made it publicly available for further analysis.

By mining the data on MARL evaluation from prior work, we highlight how certain trends, worrying inconsistencies, poor reporting, a lack of uncertainty estimation, and a general absence of proper standards for evaluation, is plaguing the current state of MARL research, making it difficult to draw sound conclusions from comparative studies. We combine these findings with the earlier issues and recommendations highlighted in the literature on RL evaluation, to propose a standardised performance evaluation protocol for cooperative MARL research.

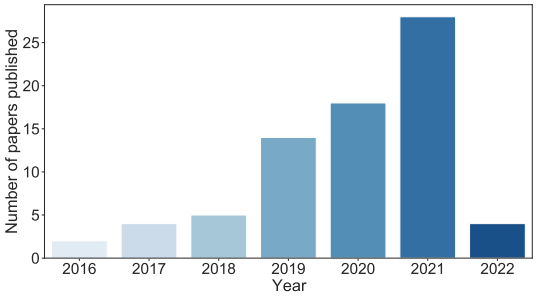

Figure 1: Recorded papers by year in the meta-analysis on evaluation methodologies in cooperative MARL.

In addition to a standardised protocol for evaluation, we expose trends in the use of benchmark environments and suggest useful standards for environment designers that could further improve the state of evaluation in MARL. We touch upon aspects of environment bias, manipulation, level/map cherry picking as well as scalability and computational considerations. Our recommendations pertain to designer specified standards concerning the use of a particular environment, its agreed upon settings, scenarios and version, as well as improved reporting by authors and preferring the use of environments designed to test generalisation.

There are clear parallels between RL and MARL evaluation, especially in the cooperative setting. Therefore, we want to emphasise that our contribution in this work is less focused on innovations in protocol design (of which much can be ported from RL) and more focused on data-driven insights on the current state of MARL research and a proposal of standards for MARL evaluation.

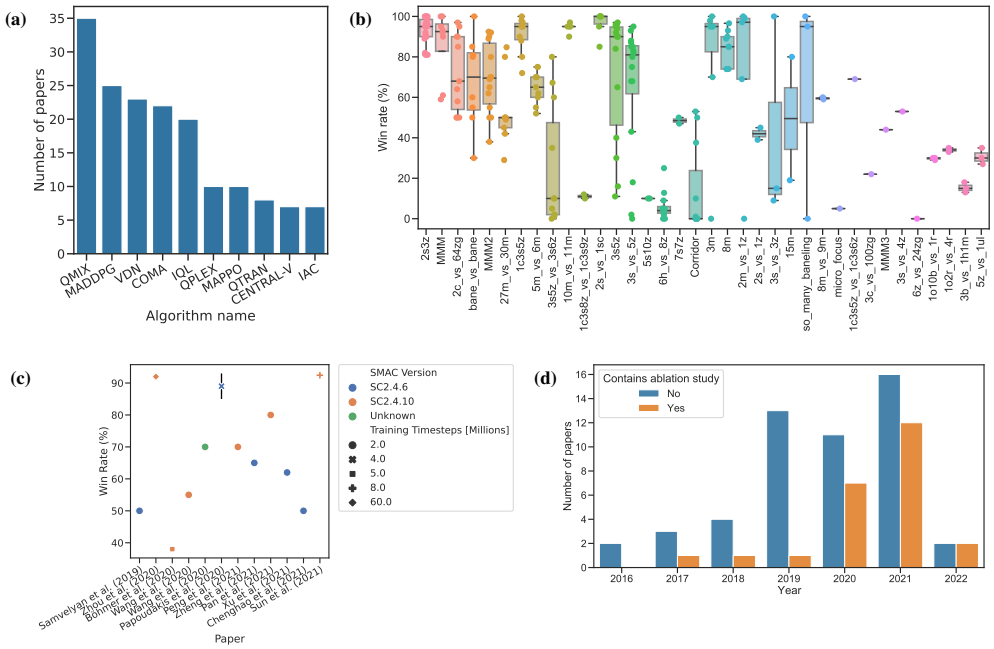

Figure 2: *Inconsistencies in performance reports and a lack of ablation studies*. **(a)** MARL algorithms ranked by popularity. **(b)** Historical performance of QMIX on different SMAC maps across papers. **(c)** Performance of QMIX on the MMM2 SMAC map as reported in different papers. **(d)** A count of papers over the years containing any type of ablation study as part of their evaluation of a newly proposed algorithm.

2 From RL to MARL evaluation: lessons, trends and recommendations

In this section, we provide a list of key lessons from RL that are applicable to MARL evaluation. We highlight important issues identified from the literature on RL evaluation, and for each issue, provide an assessment of the corresponding situation and trends in MARL. Finally, we conclude each lesson with recommendations stemming from our analysis and the literature.

2.1 Lesson 1: Know the true source of improvement and report everything

Issue in RL – *Confounding code-level optimisations and poor reporting*: It has been shown empirically that across some RL papers there is considerable variance in the reported results for the same algorithms evaluated on the same environments (Henderson et al., 2018; Jordan et al., 2020). This variance impedes the development of novel algorithmic developments by creating misleading performance comparisons and making direct comparisons of results across papers difficult. Implementation and code-level optimisation differences can have a significant impact on algorithm performance and may act as confounders during performance evaluation (Engstrom et al., 2020; Andrychowicz et al., 2020). It is rare that these implementation and code-level details are reported, or that appropriate ablations are conducted to pinpoint the true source of performance improvement.

The situation in MARL – *Similar inconsistencies and poor reporting but a promising rise in ablation studies*: There already exist some work in MARL showing the effects of specific code-level optimisations on evaluation performance (Hu et al., 2021). Employing different optimisers, exploration schedules, or simply setting the number of rollout processes to be different, can have a significant effect on algorithm performance. To better understand the variance in performance reporting across works in MARL, we focused on QMIX (Rashid et al., 2018a), the most popular algorithm in our dataset (as shown in Figure 2 (a)). In Figure 2 (b), we plot the performance of QMIX tested on different maps from the StarCraft multi-agent challenge (SMAC) environment (Samvelyan et al., 2019a), a popular benchmark in MARL. On several maps, we find wildly different reported performances with large discrepancies across papers. Although it is difficult to pin down the exact source of these differences in reports, we zoom in with our analysis to only consider a single

environment, in this case *MMM2*. We find that some of the variance is explained by differences in the environment version as well as the length of training time, as shown in Figure 2 (c). However, even when both of these aspects are controlled for, as well as any implementation or evaluation details mentioned in each paper, differences in performance are still observed (as seen by comparing orange and blue circles, respectively). This provides evidence that unreported implementation details, or differences in evaluation methodology account for some of the observed variance and act as confounders when comparing performance across papers (similar inconsistencies in other maps are shown in the Appendix). We finally consider studying the explicit attempts in published works at understanding the sources of algorithmic improvement through the use of ablation studies. We find that very few of these studies were performed in the earlier years of MARL (see Figure 2 (d)). However, even though roughly 40% of papers in 2021 still lacked any form of ablation study, we find a promising trend showing that ablation studies have become significantly more prevalent in recent years.

Recommendations – *Report all experimental details, release code and include ablation studies*: Henderson et al. (2018) emphasise that for results to be reproducible, it is important that papers report *all* experimental details. This includes hyperparameters, code-level optimisations, tuning procedures, as well as a precise description of how the evaluation was performed on both the baseline and novel work. It is also important that code be made available to easily replicate findings and stress test claims of general performance improvements. Furthermore, Engstrom et al. (2020) propose that algorithm designers be more rigorous in their analysis of the effects of individual components and how these impact performance through the use of detailed ablation studies. It is important that researchers practice diligence in attributing how the overall performance of algorithms and their underlying algorithmic behavior are affected by different proposed innovations, implementation details and code-level optimisations. In the light of our above analysis, we argue that the situation is no different in MARL, and therefore suggest the field adopt more rigorous reporting and conduct detailed ablation studies of proposed innovations.

2.2 Lesson 2: Use standardised statistical tooling for estimating and reporting uncertainty

Issue in RL – *Results in papers do not take into account uncertainty*: We have discussed how different implementations of the same algorithm with the same set of hyperparameters can lead to drastically different results. However, even under such a high degree of variability, typical methodologies often ignore the uncertainty in their reporting (Colas et al., 2018a, 2019; Jordan et al., 2020; Agarwal et al., 2022). Furthermore, most published results in RL make use of point estimates like the mean or median performance and do not take into account the statistical uncertainty arising from only using a finite number of testing runs. For instance, Agarwal et al. (2022) found that the current norm of using only a few runs to evaluate the performance of an RL algorithm is insufficient and does not account for the variability of the point estimate used. Furthermore, Agarwal et al. (2022) also revealed that the manner in which point estimates are chosen varies between authors. This inconsistency invalidates direct comparison between results across papers.

The situation in MARL – *A lack of shared standards for uncertainty estimation and concerning omissions in reporting*: In Figure 3 (a)-(c), we investigate the use of statistical aggregation and uncertainty quantification methods in MARL. We find considerable variability in the methods used, with little indication of standardisation. Perhaps more concerning is a complete absence of proper uncertainty quantification from one third of published papers. On a more positive note, we observe an upward trend in the use of standard deviation as an uncertainty measure in recent years, particularly in 2021. Furthermore, it has become fairly standard in MARL to evaluate algorithms at regular intervals during the training phase by conducting a certain number of independent evaluation runs in the environment using the current set of learned parameters. This procedure is then followed for each independent training run and results are aggregated to assess final performance. In our analysis, we find that key aspects of this procedure are regularly omitted during reporting, as shown in Figure 3 (d)-(e). Specifically, in (d), we find many papers omit details on the exact evaluation interval used, and in (e), a similar trend in omission regarding the exact number of independent evaluation runs used. Finally, in Figure 3 (f), we plot the number of independent runs used during training, showing no clear standard. Given the often high computational requirements of MARL research, it is not surprising that most works opt for a low number of independent training runs, however this remains of concern when making statistical claims.

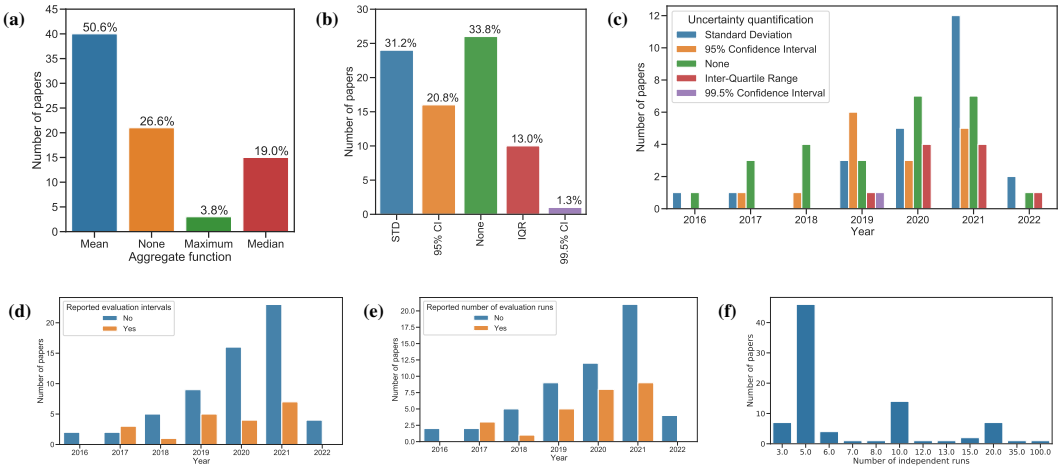

Figure 3: *Trends in performance aggregation, uncertainty quantification and omissions in reporting on key aspects of evaluation methodology*. **(a)** Distribution of performance aggregation metrics. None means that the aggregation metric was not specified in the paper. **(b)** Distribution of uncertainty methods: $\alpha\%$ confidence interval (CI), standard deviation (STD) and interquartile range (IQR). **(c)** Trends in uncertainty quantification over time. **(d)** Reporting of the evaluation interval used. **(e)** Reporting on the number of evaluation runs used. **(f)** Number of independent runs used in published work.

Recommendations – *Standardised statistical tooling and uncertainty estimation including detailed reporting*: As mentioned before, the computational requirements in MARL research often make it prohibitively difficult to run many independent experiments to properly quantify uncertainty. One approach to make sound statistical analysis more tractable, is to pool results across different tasks using the bootstrap (Efron, 1992). In particular, for RL, Agarwal et al. (2022) recommend computing stratified bootstrap confidence intervals, where instead of only using the original set of data to calculate confidence intervals, the data is resampled with replacement from M tasks, each having N runs. This process is repeated as many times as needed to approximate the sampling distribution of the statistic being calculated. Furthermore, when making a summary of overall performance across tasks it has been shown that the mean and median are insufficient, the former being dominated by outliers and the latter having higher variance. Instead, Agarwal et al. (2022) propose the use of the interquartile mean (IQM) which is more robust to outlier scores than the mean and more statistically efficient than the median. Finally, Agarwal et al. (2022) propose the use of probability of improvement scores, sample efficiency curves and performance profiles, which are commonly used to compare the performance of optimization algorithms (Dolan and Moré, 2001). These performance profiles are inherently robust to outliers on both ends of the distribution tails and allow for the comparison of relative performance at a glance. In the shared reward setting, where it is only required to track a single return value, we argue that these tools fit the exact needs of cooperative MARL, as they do in RL. Furthermore, in light of our analysis, we strongly recommend a universal standard in the use, and reporting of, evaluation parameters such as the number of independent runs, evaluation frequency, performance metrics and statistical tooling, to make comparisons across different works easier and more fair.

2.3 Lesson 3: Guard against environment misuse and overfitting

Issue in RL – *Over-tuning algorithms to perform well on a specific environment*: As early as 2011 issues with evaluation in RL came to the foreground in the form of *environment overfitting*. Whiteson et al. (2011) raised the concern that in the context of RL, researchers might over-tune algorithms to perform well on a specific benchmark at the cost of all other potential use cases. More specifically, Whiteson et al. (2011) define environment overfitting in terms of a desired target distribution. When an algorithm performs well in a specific environment but lacks performance over a target *distribution* of environments, it can be deemed to have overfit that particular environment.

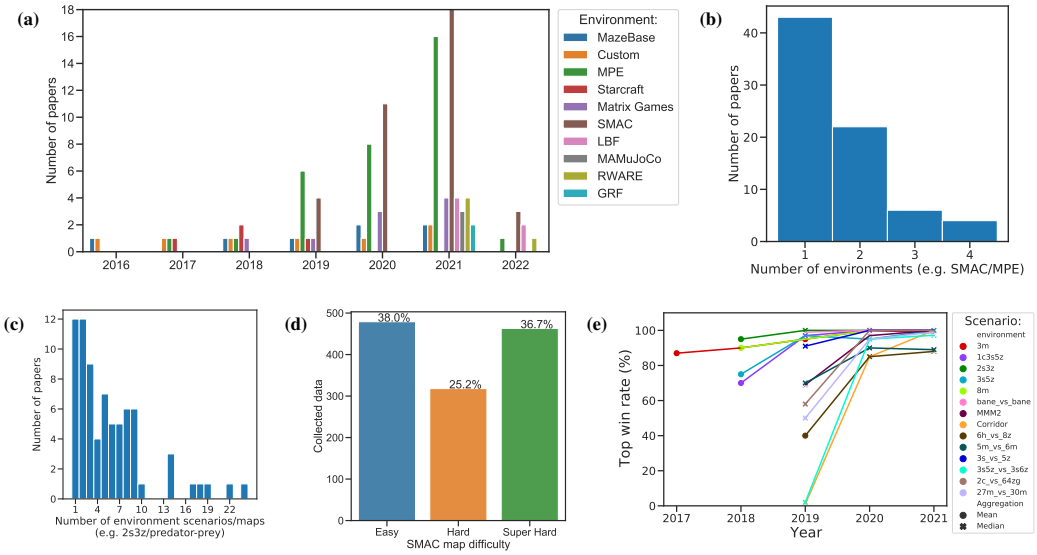

Figure 4: *Environment popularity, usage trends in papers and potential evidence of overfitting on SMAC.* **(a)** Environment adoption over time. **(b)** Number of environments used in papers. **(c)** Number of scenarios/tasks/maps used in papers. **(d)** Distribution of task difficulty of SMAC maps used in papers. **(e)** Performance trends on popular SMAC maps: Aggregation is the aggregate function used for the different reported values.

The situation in MARL – *One environment to rule them all – on the use and misuse of SMAC*: The StarCraft multi-agent challenge (SMAC) (Samvelyan et al., 2019a) has quickly risen to prominence as a key benchmark for MARL research since it's release in 2019, rivaled only in use by the multi-agent particle environment (MPE) introduced by Lowe et al. (2017a) (see Figure 4 (a)). SMAC and its accompanied MARL framework PyMARL (introduced in the same paper), fulfills several desirable properties for benchmarking: offering multiple maps of various difficulty that test key aspects of MARL algorithms and providing a unified API for running baselines as well as state-of-the-art algorithms on SMAC. Unfortunately, the wide-spread adoption of SMAC has also caused several issues, relating to environment overfitting and cherry picking of results, putting into question the credibility of claims made while using it as a benchmark.

To illustrate the above point, we start by highlighting that many MARL papers use only a single environment (e.g. SMAC or MPE) for evaluation, as shown in Figure 4 (b). This is often deemed acceptable since both SMAC and MPE provide many different tasks, or maps. For instance, in SMAC, there are 23 different maps providing a wide variety in terms of the number of agents, agent types and game dynamics. However, there is no standard, or agreed upon set of maps to use for benchmarking novel work, which makes it easy for authors to selectively subsample maps post-experiment based on the outcomes of their proposed algorithm. As shown in Figure 4 (c), although environments like SMAC and MPE offer many different testing scenarios, it is typical for papers to only use a small number of these in their reported experiments.

To concretely expose the potential danger in this author map selection bias, we redo the original analysis performed by Samvelyan et al. (2019a), using the authors' exact experimental data that was made publicly available[6], containing five independent runs for IQL (Tampuu et al., 2015a), COMA (Foerster et al., 2018a), VDN (Sunehag et al., 2017) and QMIX (Rashid et al., 2018a) on 14 SMAC maps.[7] The top row of Figure 5 shows the results of this analysis performed using the statistical tools recommended by Agarwal et al. (2022), including the probability of improvement between algorithms, performance profiles and sample efficiency curves. The results support the original claims made by Samvelyan et al. (2019a), namely that QMIX is a superior algorithm to that of VDN, COMA

[6]We applaud the authors for making their raw evaluation data publicly available. This data can be found here: https://github.com/oxwhirl/smac.

[7]The maps chosen were: 1c3s5z, 2c_vs_64zg, bane_vs_bane, MMM2, 10m_vs_11m, 27m_vs_30m, 5m_vs_6m, 2s3z, 2s_vs_1sc, s5z, 3s_vs_5z, 6h_vs_8z, 3s5z_vs_3s6z and corridor

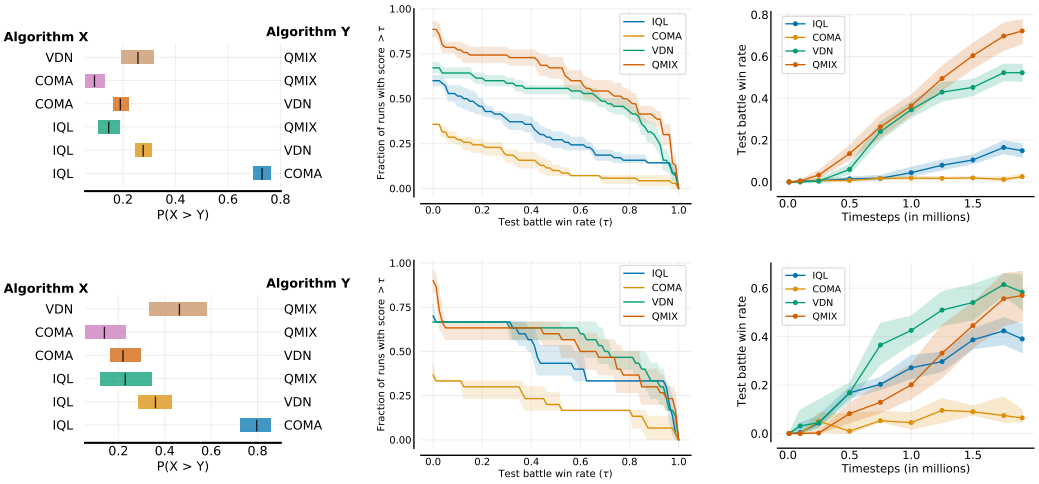

Figure 5: *Reanalysis of original SMAC experiments conducted by Samvelyan et al. (2019a) including probability of improvement, performance profiles and sample efficiency curves*. **Top row:** All 14 SMAC maps used in the original analysis. **Bottom row:** Subset of 6 maps, including 2 easy, 2 medium and 2 hard: 2s_vs_1sc, 3s_vs_5z, bane_vs_bane, 5m_vs_6m, 6h_vs_8z and corridor

and IQL, both in terms of performance and sample efficiency. However, by simply sampling a smaller set of two easy, medium and hard maps (a common spread in the literature, see Figure 4 (d)), from the original 14, giving 6 maps in total (a reasonable number according to prior work, see Figure 4 (c)), we are able to change the outcome of the analysis in support of no difference in performance between VDN and QMIX, as well as finding VDN to be more sample efficient. This is shown in the bottom row of Figure 5 and highlights the danger of a lack of standards regarding which *fixed* set of maps should be used for benchmarking.

We end our investigation into SMAC (and refer the reader to the Appendix for additional discrepancies uncovered), by looking at historical performance trends. In Figure 4 (e), we show the top win rate achieved by an algorithm in a specific year for 14 of the most popular maps used in prior work. We find that by 2021, most of these maps have converged to a win rate close or equal to 100%, while only a few maps are still situated around 80-90%. Given that many of these maps repeatedly feature across papers, and will likely be used in future work, it begs the question to what extent the MARL community has already overfit to SMAC as an evaluation benchmark.

Recommendations – *Standardised environment sets and testing for generalisation*: To solve the issue of environment overfitting, Whiteson et al. (2011) propose the use of a generalised evaluation methodology. In this approach, environments (for tuning algorithms) are freely sampled from some generalised environment set. Separately, algorithm evaluation is performed on a second set of sampled environments from the same generalised environment set, acting as a test set analogous to that used in supervised learning. Recent work in this direction include benchmarks such as Procgen (Cobbe et al., 2020), which use procedural generation to implicitly construct a distribution from which to sample test tasks. In SMAC and other environments, it is common practice to only evaluate on the exact map the algorithm was trained on, under that exact same conditions, and to not specifically test for generalisation across unseen tasks. However, many MARL algorithms are still highly sensitive to small changes in the environment and often fail to generalise to new unseen tasks (Carion et al., 2019a; Zhang et al., 2020a; Mahajan et al., 2022). This calls for more work on MARL generalisation and we recommend a stronger focus on benchmarks designed specifically to test generaliation. However, it has been noted that procedurally generated benchmarks may reduce the precision of research (Kirk et al., 2021), making it more difficult to track progress. Furthermore, MARL exibits several unique and challenging difficulties when it comes to building algorithms able to generalise, likely requiring many years of future work to surmount (Mahajan et al., 2022). Therefore, in certain cases, it might make more sense for researchers to take smaller and more precise steps towards key innovations in algorithm design by still relying on traditional environment sets. In this setting, we strongly advocate

using fixed environment sets, where ideally these are selected by the designers of each environment and are accompanied by exact instruction for their configuration, so as to be consistent across papers.

3 Towards a standardised evaluation protocol for MARL

In this section, we pool together the observations and recommendations from the previous section to provide a standardised performance evaluation protocol for cooperative MARL. We are realistic in our efforts, knowing that a single protocol is unlikely to be applicable to all MARL research. However, echoing recent work on evaluation (Ulmer et al., 2022), we stress that many of the issues highlighted previously stem from a lack of *standardisation*. Therefore, we believe a default "off-the-shelf" protocol that is able capture most settings, could provide great value to the community. If widely adopted, such a standardised protocol would make it easier and more accurate to compare across different works and remove some of the noise in the signal regarding the true rate of progress in MARL research. A summarised version of our protocol is given in the blue box at the end of this section and a concrete demonstration of its usage can be found in the Appendix.

Benchmarks and Baselines. Before giving details on the proposed protocol, we first briefly comment on benchmarks and baselines used in experiments. These choices often depend on the research question of interest, the novel work being proposed and key algorithmic capabilities to be tested. However, as alluded to in our analysis, we recommend that environment designers take full ownership regarding how their environments are to be used for evaluation. For example, if an environment has several available static tasks, the designers should specify a fixed compulsory minimum set for experiments to avoid biased subsampling by authors. It could also be helpful if designers keep track of state-of-the-art (SOTA) performances on tasks from published works and allow authors to submit these for vetting. We also strongly recommend using more than a single environment (e.g. SMAC) and preferring environments that test generalisation. Regarding baselines, we recommend that at minimum the published SOTA contender to novel work should be included. For example, if the novel proposal is a value-based off-policy algorithm for discrete environments, at minimum, it must be compared to the current SOTA value-based off-policy algorithm for discrete environments. Finally, all baselines must be tuned fairly with the same compute budget.

Evaluation parameters. In Figure 6, we show the evaluation parameters for the number of evaluation runs, the evaluation interval (top) and the training time (bottom left) used in papers. We find it is most common to use 32 evaluation episodes at every 10000 timesteps (defined as the steps taken by agents when acting in the environment) and to train for 2 million timesteps in total. We note that these numbers are skewed towards earlier years of SMAC evaluation and that recent works have since explored far longer training times. However, we argue that these longer training times are not always justified (e.g. see the bottom right of Figure 6). Furthermore, SMAC is one of the most expensive MARL environments (Appendix A.7 in Papoudakis et al. (2021)) and for the future accessibility of research in terms of scale and for fair comparisons across different works, we recommend the above commonly used values as reasonable starting defaults and support the view put forward by Dodge et al. (2019) that results should be interpreted as a function of the compute budget used. We of course recognise that these evaluation parameters can be very specific to the environment, or task, and again we urge environment designers to play a role in helping the community develop and adopt sensible standards. Finally, as done by Papoudakis et al. (2021), we recommend treating off-policy and on-policy algorithms differently and train on-policy algorithms for a factor of 10 more timesteps than off-policy algorithms due to differences in sample efficiency.

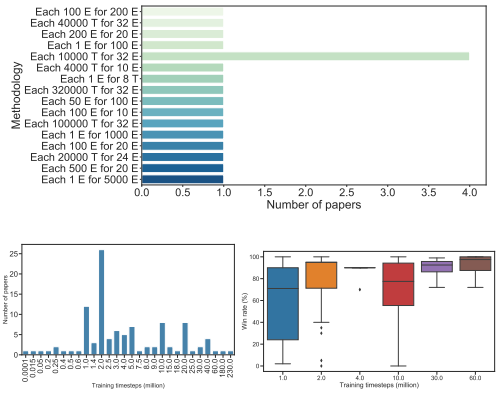

Figure 6: **Top:** Evaluation methodologies in papers: how frequent and for how many runs to do evaluation. "E" refers to episode and "T" to timestep. **Bottom:** Number of total training timesteps used in papers (left) and win rates per training timesteps on 2s3z (right).

As argued by Papoudakis et al. (2021), with modern simulators, the wall-clock time of on-policy training done in this way is typically not much slower than off-policy training.

Performance and uncertainty quantification. To aggregate performance across evaluation intervals, we recommend using the *absolute* performance metric proposed by Colas et al. (2018b), computed using the best average (over training runs) joint policy found during training. Typically, practitioners checkpoint the best performing policy parameters to use as the final model, therefore it makes sense to do evaluation in a similar way. However, to account for not averaging across different evaluation intervals, Colas et al. (2018b) recommend increasing the number of independent evaluation runs using the best policy by a factor of 10 compared to what was used at other intervals. To quantify uncertainty, we recommend using the mean with 95% confidence intervals (CIs) at each evaluation interval (computed over independent evaluation runs), and when aggregating across tasks within an environment, we recommend using the tools proposed by Agarwal et al. (2022), in particular, the inter-quartile mean (IQM) with 95% stratified Bootstrap CIs.

Reporting. We strongly recommend reporting *all* relevant experimental details including: hyper-parameters, code-level optimisations, computational requirements and frameworks used. Taking inspiration from *model cards* (Mitchell et al., 2019), we provide templates for reporting in the Appendix. Furthermore, we recommend providing experimental results in multiple formats, including plots *and* tables per task and environment as well as making all raw experimental data and code publicly available for future analysis and easy comparison. Finally, we encourage authors to include detailed ablation studies in their work, so as to be able to accurately attribute sources of improvement.

A Standardised Performance Evaluation Protocol for Cooperative MARL

Input: Environments with tasks t from a set $\mathcal{T}$. Algorithms $a \in \mathcal{A}$, including baselines and novel work.

1. Evaluation parameters – defaults

- Number of training *timesteps*, $T = 2$ million (off-policy), $T = 20$ million (on-policy).
- Number of independent training *runs*, $R = 10$ (from Agarwal et al. (2022))
- Number of independent evaluation *episodes* per interval, $E = 32$.
- Evaluation *intervals*, $i \in \mathcal{I}$, at every 10000 timesteps.

2. Performance and uncertainty quantification

1. Performance metric: Always use returns G (applicable to all environments), *and* the environment specific metric (e.g. Win rate).
2. Per task evaluation: Compute the mean G_t^a over E episodes at each evaluation interval i, where G_t^a is the return of algorithm a on task t, with 95% CI, for all a.
3. Per environment evaluation:
   - Compute the normalised *absolute* return (Colas et al., 2018b) as the mean return of $10 \times E = 320$ evaluation episodes using the best joint policy found during training and normalising the return to be in the range $[0, 1]$ using $(G_t^a - \min(G_t))/(\max(G_t) - \min(G_t))$, where G_t is the return for all algorithms on task t.
   - For each algorithm a, form an evaluation matrix with shape $(R, |\mathcal{T}|)$ where each entry is the normalised absolute return for a specific training run on a specific task.
   - Compute the IQM and optimality gap with 95% stratified Bootstrap CIs, probability of improvement scores and performance profiles, to compare the algorithms, using the tools proposed by Agarwal et al. (2022).[a] Sample efficiency curves can be computed by using normalised returns at each evaluation interval.

3. Reporting

- Experiments: All hyperparameters, code-level optimisations, computational requirements and framework details.
- Plots: All task and environment evaluations as well as ablation study results.
- Tables: Normalised absolute performance per task with 95% CI for all tasks, IQM with 95% stratified Bootstrap CIs per environment for all environments.
- Public repository: Raw evaluation data and code implementations.

[a] These can be found in the `rliable` library: https://github.com/google-research/rliable

4 Conclusion and future work

In this work, we argue for the power of standardisation. In a fast-growing field such as MARL, it becomes ever more important to be able to dispel illusions of rapid progress, potentially misleading the field and resulting in wasted time and effort. We hope to break the spell by proposing a sensible standardised performance evaluation protocol, motivated in part by the literature on evaluation in RL, as well as by a meta-analysis of prior work in MARL. If widely adopted, such a protocol could make comparisons across different works much faster, easier and more accurate. However, certain aspects of evaluation are better left standardised outside of the control or influence of authors, such as protocols pertaining to the use of benchmark environments. We believe this is an overlooked issue and an important area for future work by the community, and specifically environment designers, to jointly establish better standards and protocols for evaluation and environment use. Finally, we encourage the community to move beyond the use of only one or two environments with static task sets (e.g. MPE and SMAC) and focus more on building algorithms, environments and tools for improving generalisation in MARL.

A clear limitation of our work is our focus on the cooperative setting. Interesting works have developed protocols and environments for evaluation in both the competitive and mixed settings (Omidshafiei et al., 2019; Rowland et al., 2019; Leibo et al., 2021). We find this encouraging and argue for similar efforts in the adoption of proposed standards for evaluation.

Acknowledgments and Disclosure of Funding

The authors would like to kindly thank the following people for useful discussions and feedback on this work: Jonathan Shock, Matthew Morris, Claude Formanek, Asad Jeewa, Kale-ab Tessera, Sinda Ben Salem, Khalil Gorsan Mestiri, Chaima Wichka and Sasha Abramowitz. Also, Roland Dubb acknowledges support from the University of Cape Town and ETDP SETA.

References

R. Islam, P. Henderson, M. Gomrokchi, and D. Precup, "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control," 2017. [Online]. Available: https://arxiv.org/abs/1708.04133

M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling, "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," 2017. [Online]. Available: https://arxiv.org/abs/1709.06009

P. Henderson, "Reproducibility and reusability in deep reinforcement learning," Master's thesis, McGill University, 2018.

A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," 2018. [Online]. Available: https://arxiv.org/abs/1806.07937

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning that Matters," 2018.

C. Colas, O. Sigaud, and P.-Y. Oudeyer, "How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments," 2018.

——, "A Hitchhiker's Guide to Statistical Comparisons of Reinforcement Learning Algorithms," 2019.

S. C. Y. Chan, S. Fishman, J. Canny, A. Korattikara, and S. Guadarrama, "Measuring the Reliability of Reinforcement Learning Algorithms," 2020.

S. M. Jordan, Y. Chandak, D. Cohen, M. Zhang, and P. S. Thomas, "Evaluating the Performance of Reinforcement Learning Algorithms," 2020.

L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep policy gradients: A case study on ppo and trpo," 2020.

R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, "Deep Reinforcement Learning at the Edge of the Statistical Precipice," 2022.

J. N. Foerster, "Deep multi-agent reinforcement learning," Ph.D. dissertation, University of Oxford, 2018.

A. Oroojlooyjadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *CoRR*, vol. abs/1908.03963, 2019. [Online]. Available: http://arxiv.org/abs/1908.03963

Y. Yang and J. Wang, "An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective," *CoRR*, vol. abs/2011.00583, 2020. [Online]. Available: https://arxiv.org/abs/2011.00583

C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, "The surprising effectiveness of MAPPO in cooperative, multi-agent games," *CoRR*, vol. abs/2103.01955, 2021. [Online]. Available: https://arxiv.org/abs/2103.01955

G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. [Online]. Available: http://arxiv.org/abs/2006.07869

J. Hu, S. Jiang, S. A. Harding, H. Wu, and S. wei Liao, "Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning," 2022.

M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski *et al.*, "What matters in on-policy reinforcement learning? a large-scale empirical study," *arXiv preprint arXiv:2006.05990*, 2020.

J. Hu, S. Jiang, S. A. Harding, H. Wu, and S. wei Liao, "Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning," 2021.

T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.

M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft Multi-Agent Challenge," *CoRR*, vol. abs/1902.04043, 2019.

B. Efron, *Bootstrap Methods: Another Look at the Jackknife*. New York, NY: Springer New York, 1992, pp. 569–593. [Online]. Available: https://doi.org/10.1007/978-1-4612-4380-9_41

E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," 2001. [Online]. Available: https://arxiv.org/abs/cs/0102001

S. Whiteson, B. Tanner, M. E. Taylor, and P. Stone, "Protecting against evaluation overfitting in empirical reinforcement learning," in *2011 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*. IEEE, 2011, pp. 120–127.

R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," 2015.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

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

K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 2048–2056.

N. Carion, N. Usunier, G. Synnaeve, and A. Lazaric, "A structured prediction approach for generalization in cooperative multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

T. Zhang, H. Xu, X. Wang, Y. Wu, K. Keutzer, J. E. Gonzalez, and Y. Tian, "Multi-agent collaboration via reward attribution decomposition," *arXiv preprint arXiv:2010.08531*, 2020.

A. Mahajan, M. Samvelyan, T. Gupta, B. Ellis, M. Sun, T. Rocktäschel, and S. Whiteson, "Generalization in cooperative multi-agent systems," *arXiv preprint arXiv:2202.00104*, 2022.

R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of generalisation in deep reinforcement learning," *arXiv preprint arXiv:2111.09794*, 2021.

D. Ulmer, E. Bassignana, M. Müller-Eberstein, D. Varab, M. Zhang, C. Hardmeier, and B. Plank, "Experimental standards for deep learning research: A natural language processing perspective," *arXiv preprint arXiv:2204.06251*, 2022.

J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith, "Show your work: Improved reporting of experimental results," *arXiv preprint arXiv:1909.03004*, 2019.

C. Colas, O. Sigaud, and P.-Y. Oudeyer, "Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms," in *International conference on machine learning*. PMLR, 2018, pp. 1039–1048.

M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 220–229.

S. Omidshafiei, C. Papadimitriou, G. Piliouras, K. Tuyls, M. Rowland, J.-B. Lespiau, W. M. Czarnecki, M. Lanctot, J. Perolat, and R. Munos, "α-rank: Multi-agent evaluation by evolution," *Scientific reports*, vol. 9, no. 1, pp. 1–29, 2019.

M. Rowland, S. Omidshafiei, K. Tuyls, J. Perolat, M. Valko, G. Piliouras, and R. Munos, "Multiagent evaluation under incomplete information," 2019. [Online]. Available: https://arxiv.org/abs/1909.09849

J. Z. Leibo, E. Duéñez-Guzmán, A. S. Vezhnevets, J. P. Agapiou, P. Sunehag, R. Koster, J. Matyas, C. Beattie, I. Mordatch, and T. Graepel, "Scalable Evaluation of Multi-Agent Reinforcement Learning with Melting Pot," 2021.

S. Sukhbaatar, a. szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf

J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf

S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *ICML*, 2017, pp. 2681–2690. [Online]. Available: http://proceedings.mlr.press/v70/omidshafiei17a.html

R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NIPS*, 2017, pp. 6382–6393. [Online]. Available: http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments

J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1146–1155. [Online]. Available: https://proceedings.mlr.press/v70/foerster17b.html

E. Wei, D. Wicke, D. Freelan, and S. Luke, "Multiagent soft q-learning," in *2018 AAAI Spring Symposia, Stanford University, Palo Alto, California, USA, March 26-28, 2018*. AAAI Press, 2018. [Online]. Available: https://aaai.org/ocs/index.php/SSS/SSS18/paper/view/17508

J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI*. AAAI Press, 2018, pp. 2974–2982.

P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, E. André, S. Koenig, M. Dastani, and G. Sukthankar, Eds. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018, pp. 2085–2087. [Online]. Available: http://dl.acm.org/citation.cfm?id=3238080

T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4295–4304. [Online]. Available: https://proceedings.mlr.press/v80/rashid18a.html

A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rye7knCqK7

S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2961–2970. [Online]. Available: https://proceedings.mlr.press/v97/iqbal19a.html

S. Q. Zhang, Q. Zhang, and J. Lin, "Efficient communication in multi-agent reinforcement learning via variance based control," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/14cfdb59b5bda1fc245aadae15b1984a-Paper.pdf

A. Malysheva, D. Kudenko, and A. Shpilman, "Magnet: Multi-agent graph network for deep multi-agent reinforcement learning," in *2019 XVI International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)*, 2019, pp. 171–176.

H. Mao, Z. Zhang, Z. Xiao, and Z. Gong, "Modelling the dynamic joint policy of teammates with attention multi-agent DDPG," in *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1108–1116.

M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. N. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," in *AAMAS*, 2019, pp. 2186–2188. [Online]. Available: http://dl.acm.org/citation.cfm?id=3332052

N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas, "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 3040–3049. [Online]. Available: https://proceedings.mlr.press/v97/jaques19a.html

Y. Du, L. Han, M. Fang, J. Liu, T. Dai, and D. Tao, "Liir: Learning individual intrinsic reward in multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/07a9d3fed4c5ea6b17e80258dee231fa-Paper.pdf

A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "Maven: Multi-agent variational exploration," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/f816dc0acface7498e10496222e9db10-Paper.pdf

C. Schroeder de Witt, J. Foerster, G. Farquhar, P. Torr, W. Boehmer, and S. Whiteson, "Multi-agent common knowledge reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/f968fdc88852a4a3a27a81fe3f57bfc5-Paper.pdf

N. Carion, N. Usunier, G. Synnaeve, and A. Lazaric, "A structured prediction approach for generalization in cooperative multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/3c3c139bd8467c1587a41081ad78045e-Paper.pdf

A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "TarMAC: Targeted multi-agent communication," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 1538–1546. [Online]. Available: https://proceedings.mlr.press/v97/das19a.html

K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 5887–5896. [Online]. Available: https://proceedings.mlr.press/v97/son19a.html

T. Wang, J. Wang, Y. Wu, and C. Zhang, "Influence-based multi-agent exploration," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJgy96EYvr

Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 7211–7218, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6211

J. Ma and F. Wu, "Feudal multi-agent deep reinforcement learning for traffic signal control," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 816–824.

I.-J. Liu, R. A. Yeh, and A. G. Schwing, "Pic: Permutation invariant critic for multi-agent deep reinforcement learning," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 590–602. [Online]. Available: https://proceedings.mlr.press/v100/liu20a.html

W. Wang, T. Yang, Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, and Y. Gao, "Action semantics network: Considering the effects of actions in multiagent systems," in *ICLR*, 2020. [Online]. Available: https://openreview.net/forum?id=ryg48p4tPH

S. Q. Zhang, Q. Zhang, and J. Lin, "Succinct and robust multi-agent communication with temporal message control," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 17 271–17 282. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c82b013313066e0702d58dc70db033ca-Paper.pdf

J. Xu, F. Zhong, and Y. Wang, "Learning multi-agent coordination for enhancing target coverage in directional sensor networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 10 053–10 064. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/7250eb93b3c18cc9daa29cf58af7a004-Paper.pdf

T. Wang, J. Wang, C. Zheng, and C. Zhang, "Learning nearly decomposable value functions via communication minimization," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJx-3grYDB

J. Roy, P. Barde, F. Harvey, D. Nowrouzezahrai, and C. Pal, "Promoting coordination through policy regularization in multi-agent deep reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 15 774–15 785. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/b628386c9b92481fab68fbf284bd6a64-Paper.pdf

J. Wang, Y. Zhang, T.-K. Kim, and Y. Gu, "Shapley q-value: A local reward approach to solve global reward games," in *AAAI*, 2020, pp. 7285–7292. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/6220

W. Boehmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 980–991. [Online]. Available: https://proceedings.mlr.press/v119/boehmer20a.html

Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, and X. Wang, "Evolutionary population curriculum for scaling multi-agent reinforcement learning," in *ICLR*, 2020.

F. Christianos, L. Schäfer, and S. Albrecht, "Shared experience actor-critic for multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 10 707–10 717. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf

C. Wen, X. Yao, Y. Wang, and X. Tan, "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, 2020, pp. 7301–7308. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6223

A. Agarwal, S. Kumar, K. P. Sycara, and M. Lewis, "Learning transferable cooperative behavior in multi-agent teams," in *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1741–1743.

G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Comparative evaluation of multi-agent deep reinforcement learning algorithms," vol. abs/2006.07869, 2020. [Online]. Available: https://arxiv.org/abs/2006.07869

Z. Ding, T. Huang, and Z. Lu, "Learning individually inferred communication for multi-agent cooperation," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 22 069–22 079. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/fb2fcd534b0ff3bbed73cc51df620323-Paper.pdf

H. Hu and J. N. Foerster, "Simplified action decoder for deep multi-agent reinforcement learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1xm3RVtwB

M. Zhou, Z. Liu, P. Sui, Y. Li, and Y. Y. Chung, "Learning implicit credit assignment for cooperative multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 11 853–11 864. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/8977ecbb8cb82d77fb091c7a7f186163-Paper.pdf

J. Chen, Y. Zhang, Y. Xu, H. Ma, H. Yang, J. Song, Y. Wang, and Y. Wu, "Variational automatic curriculum learning for sparse-reward cooperative multi-agent problems," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 9681–9693. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/503e7dbbd6217b9a591f3322f39b5a6c-Paper.pdf

M. Chen, Y. Li, E. Wang, Z. Yang, Z. Wang, and T. Zhao, "Pessimism meets invariance: Provably efficient offline mean-field multi-agent rl," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 17 913–17 926. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/9559fc73b13fa721a816958488a5b449-Paper.pdf

S. Li, J. K. Gupta, P. Morales, R. Allen, and M. J. Kochenderfer, "Deep implicit coordination graphs for multi-agent reinforcement learning," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '21. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2021, p. 764–772.

W.-F. Sun, C.-K. Lee, and C.-Y. Lee, "Dfac framework: Factorizing the value function via quantile mixture for multi-agent distributional q-learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 9945–9954. [Online]. Available: https://proceedings.mlr.press/v139/sun21c.html

F. Christianos, G. Papoudakis, M. A. Rahman, and S. V. Albrecht, "Scaling multi-agent reinforcement learning with selective parameter sharing," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 1989–1998. [Online]. Available: https://proceedings.mlr.press/v139/christianos21a.html

J. Wang, Z. Ren, B. Han, J. Ye, and C. Zhang, "Towards understanding cooperative multi-agent q-learning with value factorization," in *Advances in Neural Information Processing*

Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 29 142–29 155. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/f3f1fa1e4348bfbebdeee8c80a04c3b9-Paper.pdf

K. M. Lee, S. G. Subramanian, and M. Crowley, "Investigation of independent reinforcement learning algorithms in multi-agent environments," in *Deep RL Workshop NeurIPS 2021*, 2021. [Online]. Available: https://openreview.net/forum?id=8MkKGZ2AlmJ

L. Chenghao, T. Wang, C. Wu, Q. Zhao, J. Yang, and C. Zhang, "Celebrating diversity in shared multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 3991–4002. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/20aee3a5f4643755a79ee5f6a73050ac-Paper.pdf

T. Wang, T. Gupta, A. Mahajan, B. Peng, S. Whiteson, and C. Zhang, "Rode: Learning roles to decompose multi-agent tasks," in *ICLR*, 2021. [Online]. Available: https://openreview.net/forum?id=TTUVg6vkNjK

Y. Xiao, X. Lyu, and C. Amato, "Local advantage actor-critic for robust multi-agent deep reinforcement learning," in *MRS*. IEEE, 2021, pp. 155–163.

Z. Xu, D. Li, Y. Bai, and G. Fan, "MMD-MIX: value function factorisation with maximum mean discrepancy for cooperative multi-agent reinforcement learning," in *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. IEEE, 2021, pp. 1–7. [Online]. Available: https://doi.org/10.1109/IJCNN52387.2021.9533636

J. Jiang and Z. Lu, "The emergence of individuality," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 4992–5001. [Online]. Available: https://proceedings.mlr.press/v139/jiang21g.html

P. Leroy, D. Ernst, P. Geurts, G. Louppe, J. Pisane, and M. Sabatelli, "QVMix and QVMix-Max: Extending the Deep Quality-Value Family of Algorithms to Cooperative Multi-Agent Reinforcement Learning," in *Proceedings of the AAAI-21 Workshop on Reinforcement Learning in Games*, 2021. [Online]. Available: https://arxiv.org/abs/2012.12062

T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning." NeurIPS, 2021.

J. Su, S. C. Adams, and P. A. Beling, "Value-decomposition multi-agent actor-critics," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 11 352–11 360. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17353

L. Pan, T. Rashid, B. Peng, L. Huang, and S. Whiteson, "Regularized softmax deep multi-agent q-learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 1365–1377. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/0a113ef6b61820daa5611c870ed8d5ee-Paper.pdf

I.-J. Liu, U. Jain, R. A. Yeh, and A. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 6826–6836. [Online]. Available: https://proceedings.mlr.press/v139/liu21j.html

I. Saeed, A. C. Cullen, S. M. Erfani, and T. Alpcan, "Domain-aware multiagent reinforcement learning in navigation," in *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. IEEE, 2021, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN52387.2021.9533975

B. Guresti and N. K. Ure, "Evaluating generalization and transfer capacity of multi-agent reinforcement learning across variable number of agents," *CoRR*, vol. abs/2111.14177, 2021. [Online]. Available: https://arxiv.org/abs/2111.14177

L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiosity-driven exploration," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 3757–3769. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/1e8ca836c962598551882e689265c1c5-Paper.pdf

E. Marchesini and A. Farinelli, "Centralizing state-values in dueling networks for multi-robot reinforcement learning mapless navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*. IEEE, 2021, pp. 4583–4588. [Online]. Available: https://doi.org/10.1109/IROS51168.2021.9636349

J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "{QPLEX}: Duplex dueling multi-agent q-learning," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=Rcmk0xxIQV

J. G. Kuba, M. Wen, L. Meng, s. gu, H. Zhang, D. Mguni, J. Wang, and Y. Yang, "Settling the variance of multi-agent policy gradients," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 13 458–13 470. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/6fe6a8a6e6cb710584efc4af0c34ce50-Paper.pdf

B. Peng, T. Rashid, C. Schroeder de Witt, P.-A. Kamienny, P. Torr, W. Boehmer, and S. Whiteson, "Facmac: Factored multi-agent centralised policy gradients," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 12 208–12 221. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/65b9eea6e1cc6bb9f0cd2a47751a186f-Paper.pdf

L. Yuan, J. Wang, F. Zhang, C. Wang, Z. Zhang, Y. Yu, and C. Zhang, "Multi-agent incentive communication via decentralized teammate modeling," 2022.

D. H. Mguni, T. Jafferjee, J. Wang, N. Perez-Nieves, O. Slumbers, F. Tong, Y. Li, J. Zhu, Y. Yang, and J. Wang, "LIGS: Learnable intrinsic-reward generation selection for multi-agent learning," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=CpTuR2ECuW

Y. Wang, fangwei zhong, J. Xu, and Y. Wang, "Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=2t7CkQXNpuq

J. G. Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, and Y. Yang, "Trust region policy optimisation in multi-agent reinforcement learning," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=EcGGFkNTxdJ

S. A. Stavroulakis and B. Sengupta, "Reinforcement learning for location-aware warehouse scheduling," in *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. [Online]. Available: https://openreview.net/forum?id=Bt-gaVaVJ-9

A. Castagna and I. Dusparic, "Multi-agent transfer learning in reinforcement learning-based ride-sharing systems," in *Proceedings of the 14th International Conference on Agents and Artificial Intelligence, ICAART 2022, Volume 2, Online Streaming, February 3-5, 2022*, A. P. Rocha, L. Steels, and H. J. van den Herik, Eds. SCITEPRESS, 2022, pp. 120–130. [Online]. Available: https://doi.org/10.5220/0010785200003116

M. Zawalski, B. Osinski, H. Michalewski, and P. Milos, "Off-policy correction for multi-agent reinforcement learning," in *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022, pp. 1774–1776.

R. Avalos, M. Reymond, A. Nowé, and D. M. Roijers, "Local advantage networks for cooperative multi-agent reinforcement learning," in *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022, pp. 1524–1526.

Y. X. Xueguang Lyu, "A deeper understanding of state-based critics in multi-agent reinforcement learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. [Online]. Available: https://par.nsf.gov/biblio/10315765

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016. [Online]. Available: https://arxiv.org/abs/1602.01783

A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PLOS ONE*, vol. 12, 11 2015.

C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," 2021. [Online]. Available: https://arxiv.org/abs/2103.01955

Y. J. Park, Y. J. Lee, and S. B. Kim, "Cooperative multi-agent reinforcement learning with approximate model learning," *IEEE Access*, vol. 8, pp. 125 389–125 400, 2020.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

S. Mohanty, E. Nygren, F. Laurent, M. Schneider, C. Scheller, N. Bhattacharya, J. Watson, A. Egli, C. Eichenberger, C. Baumberger *et al.*, "Flatland-rl: Multi-agent reinforcement learning on trains," *arXiv preprint arXiv:2012.05893*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We did this both in our introduciton and conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We did not find this necessary given the nature of the work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [No]
   (b) Did you include complete proofs of all theoretical results? [No]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In main text and for experiments in supplementary material.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] Links to repositories were given where the license can be found.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] All collected data is publicaly available, accesilble via open-access publication.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The meta-analysis data contains information typical to any research meta-analysis, including author information, paper titles and information extracted from each paper.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Data collection and annotation methodology

This section outlines the search methodology and data recording practices used to collect the dataset of algorithm performance and evaluation methodologies for the field of cooperative MARL.[8] The dataset used in the main body of this paper reflects the algorithm evaluation practices of published cooperative MARL papers only. We note that the original data collection was not restricted to accepted publications and cooperative MARL, as it instead attempts to incorporate all prominent and contemporary deep MARL algorithms and approaches from all available studies. This is reflected in this appendix, where we refer to data collected from all recorded papers (published, rejected, unknown, and non-cooperative) as *all papers*. Similarly, we refer to the data collected from cooperative published papers (which were used in the main body of this work) as *the main papers*. The non-published papers and non-cooperative published papers are referred to as *the other papers*.

A.1 Paper search strategy

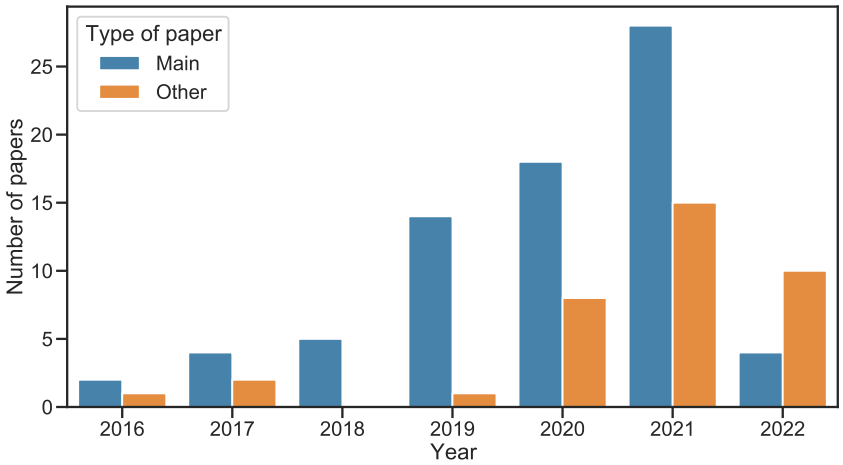

Figure 7: Recorded papers by year

In order to gather data on MARL algorithm performance evaluation, we gathered relevant MARL research papers which were published between the years 2016 and 2022. To identify relevant studies, we searched for relevant research key terms, such as "Multi-agent RL", "MARL evaluation" and "Benchmarking MARL". We searched the arXiv website for these terms in different combinations of the title, abstract, and keywords. Additionally, several papers were included from the reference list of other papers. Although we do not claim to have a dataset comprised of all modern deep MARL algorithms, we strive to collect data on at least all of the most widely used deep MARL algorithms. To our knowledge, all major deep MARL algorithms are represented in our dataset and this dataset is the first of its kind. The search queries were finalized on the 8th of April 2022. The published research papers that we recorded can be found in Table 1, where these were published at various conferences including ICML, NeurIPS, ICLR, and others.

A.2 Filtering data to find relevant studies

Following the initial data collection, the dataset was refined to ensure relevance using the following criteria:

- The papers must be either peer reviewed conference or journal papers, and published in the English language.
- Papers were restricted to only those which focus exclusively on the cooperative MARL case.

[8] Meta-analysis dataset on MARL evaluation https://bit.ly/3LpxAMb

Table 1: Published cooperative MARL research papers collected and manually annotated for data analysis of algorithm performance evaluation methods.

Title	Authors	Conference
Learning Multiagent Communication with Backpropagation	Sukhbaatar et al. (2016)	NeurIPS
Learning to Communication in Deep Multi-Agent Reinforcement Learning	Foerster et al. (2016)	NeurIPS
Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability	Omidshafiei et al. (2017)	ICML
Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments	Lowe et al. (2017b)	NeurIPS
Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning	Foerster et al. (2017)	ICML
MultiAgent Soft-Q Learning	Wei et al. (2018)	AAAI
Counterfactual Multi-Agent Policy Gradients	Foerster et al. (2018b)	AAAI
Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward	Sunehag et al. (2018)	AAMAS
QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning	Rashid et al. (2018b)	ICML
Learning when to Communicate at Scale in Multiagent Cooperative and Competitive Tasks	Singh et al. (2019)	ICLR
Actor-Attention-Critic for Multi-Agent Reinforcement Learning	Iqbal and Sha (2019)	ICML
Efficient Communication in Multi-Agent Reinforcement Learning via Variance Based Control	Zhang et al. (2019)	NeurIPS
MAGNet: Multi-agent Graph Network for Deep Multi-agent Reinforcement Learning	Malysheva et al. (2019)	IEEE
Modelling the Dynamic Joint Policy of Teammates with Attention Multi-agent DDPG	Mao et al. (2019)	AAMAS
The StarCraft Multi-Agent Challenge	Samvelyan et al. (2019b)	AAMAS
Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning	Jaques et al. (2019)	ICML
LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning	Du et al. (2019)	NeurIPS
MAVEN: Multi-Agent Variational Exploration	Mahajan et al. (2019)	NeurIPS
Multi-Agent Common Knowledge Reinforcement Learning	Schroeder de Witt et al. (2019)	NeurIPS
A Structured Prediction Approach for Generalization in Cooperative Multi-Agent Reinforcement Learning	Carion et al. (2019b)	NeurIPS
TarMAC: Targeted Multi-Agent Communication	Das et al. (2019)	ICML
QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning	Son et al. (2019)	ICML
Influence-Based Multi-Agent Exploration	Wang et al. (2020a)	ICLR
Multi-Agent Game Abstraction via Graph Attention Neural Network	Liu et al. (2020a)	AAAI
Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning	Ma and Wu (2020)	AAMAS
PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning	Liu et al. (2020b)	CoRL
Action Semantics Network: Considering the Effects of Actions in Multiagent Systems	Wang et al. (2020b)	ICLR
Succinct and Robust Multi-Agent Communication With Temporal Message Control	Zhang et al. (2020b)	NeurIPS
Learning Multi-Agent Coordination for Enhancing Target Coverage in Directional Sensor Networks	Xu et al. (2020)	NeurIPS
Learning Nearly Decomposable Value Functions Via Communication Minimization	Wang et al. (2020c)	ICLR
Promoting Coordination through Policy Regularization in Multi-Agent Deep Reinforcement Learning	Roy et al. (2020)	NeurIPS
Shapley Q-value: A Local Reward Approach to Solve Global Reward Games	Wang et al. (2020d)	AAAI
Deep Coordination Graphs	Boehmer et al. (2020)	ICML
Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning	Long et al. (2020)	ICLR
Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning	Christianos et al. (2020)	NeurIPS
SMIX(λ): Enhancing Centralized Value Functions for Cooperative Multi-Agent Reinforcement Learning	Wen et al. (2020)	AAAI
Learning Transferable Cooperative Behavior in Multi-Agent Teams	Agarwal et al. (2020)	AAMAS
Comparative Evaluation of Cooperative Multi-Agent Deep Reinforcement Learning Algorithms	Papoudakis et al. (2020)	AAMAS
Learning Individually Inferred Communication for Multi-Agent Cooperation	Ding et al. (2020)	NeurIPS
Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning	Hu and Foerster (2020)	ICLR
Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning	Zhou et al. (2020)	NeurIPS
Variational Automatic Curriculum Learning for Sparse-Reward Cooperative Multi-Agent Problems	Chen et al. (2021a)	NeurIPS
Pessimism Meets Invariance: Provably Efficient Offline Mean-Field Multi-Agent RL	Chen et al. (2021b)	NeurIPS
Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning	Li et al. (2021)	AAMAS
DFAC Framework: Factorizing the Value Function via Quantile Mixture for Multi-Agent Distributional Q-Learning	Sun et al. (2021)	ICML
Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing	Christianos et al. (2021)	ICML
Towards Understanding Cooperative Multi-Agent Q-Learning with Value Factorization	Wang et al. (2021a)	NeurIPS
Investigation of Independent Reinforcement Learning Algorithms in Multi-Agent Environments	Lee et al. (2021)	NeurIPS
Celebrating Diversity in Shared Multi-Agent Reinforcement Learning	Chenghao et al. (2021)	NeurIPS
RODE: Learning Roles to Decompose Multi-Agent Tasks	Wang et al. (2021b)	ICLR
Local Advantage Actor-Critic for Robust Multi-Agent Deep Reinforcement Learning	Xiao et al. (2021)	IEEE MRS
MMD-MIX: Value Function Factorisation with Maximum Mean Discrepancy for Cooperative Multi-Agent Reinforcement Learning	Xu et al. (2021)	IJCNN
The Emergence of Individuality	Jiang and Lu (2021)	ICML
QVMix and QVMix-Max: Extending the Deep Quality-Value Family of Algorithms to Cooperative Multi-Agent Reinforcement Learning	Leroy et al. (2021)	AAAI
Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning	Rashid et al. (2021)	NeurIPS
Value-Decomposition Multi-Agent Actor-Critics	Su et al. (2021)	AAAI
Regularized Softmax Deep Multi-Agent Q-Learning	Pan et al. (2021)	NeurIPS
Cooperative Exploration for Multi-Agent Deep Reinforcement Learning	Liu et al. (2021)	ICML
Domain-Aware Multiagent Reinforcement Learning in Navigation	Saeed et al. (2021)	IJCNN
Evaluating Generalization and Transfer Capacity of Multi-Agent Reinforcement Learning Across Variable Number of Agents	Guresti and Ure (2021)	AAAI
Episodic Multi-agent Reinforcement Learning with Curiosity-driven Exploration	Zheng et al. (2021)	NeurIPS
Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks	Papoudakis et al. (2021)	NeurIPS
Centralizing State-Values in Dueling Networks for Multi-Robot Reinforcement Learning Mapless Navigation	Marchesini and Farinelli (2021)	IROS
QPLEX: Duplex Dueling Multi-Agent Q-Learning	Wang et al. (2021c)	ICLR
Settling the Variance of Multi-Agent Policy Gradients	Kuba et al. (2021)	NeurIPS
FACMAC: Factored Multi-Agent Centralised Policy Gradients	Peng et al. (2021)	NeurIPS
Multi-Agent Incentive Communication via Decentralized Teammate Modeling	Yuan et al. (2022)	AAAI
LIGS: Learnable Intrinsic-Reward Generation Selection for Multi-Agent Learning	Mguni et al. (2022)	ICLR
ToM2C: Target-oriented Multi-agent Communication and Cooperation with Theory of Mind	Wang et al. (2022)	ICLR
Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning	Kuba et al. (2022)	ICLR
Reinforcement Learning for Location-Aware Warehouse Scheduling	Stavroulakis and Sengupta (2022)	ICLR
Multi-agent Transfer Learning in Reinforcement Learning-based Ride-sharing Systems	Castagna and Dusparic (2022)	ICAART
Off-Policy Correction For Multi-Agent Reinforcement Learning	Zawalski et al. (2022)	AAMAS
Local Advantage Networks for Cooperative Multi-Agent Reinforcement Learning	Avalos et al. (2022)	AAMAS
A Deeper Understanding of State-Based Critics in Multi-Agent Reinforcement Learning	Xueguang Lyu (2022)	AAAI

A.3 Annotations

The collected dataset was manually annotated to record methods of algorithm performance evaluation. The dataset records the algorithms, environments, and tasks used as well as all aspects relating to the algorithm performance evaluation procedure that was available from the papers. The following provides further details on the data annotation procedure:

A1. The names and dates of papers are recorded along with the conferences they are published into and research labs associated with the authors.

A2. The algorithms being evaluated are recorded. In some cases the paper-specific names of algorithms have been appropriately adapted. This is in cases where uniquely named algorithms have only minor differences from their baselines. Further details of this standardisation appear in subsection A.3.2. The algorithm libraries used are recorded when applicable (e.g. EPyMARL Papoudakis et al. (2021)).

A3. We recorded the MARL environments, their sub-tasks/maps/scenarios and the choice of version used for evaluation. Environment sub-tasks with different names, but which refer to the identical sub-tasks were given standardised names (e.g. cooperative communication is the second name for Speaker-Listener task in MPE).

A4. With regard to performance measurement, we recorded the aggregation functions across runs or episodes (e.g. means) and, the metrics used (e.g. SMAC win rates or max rewards) along with their measure of spread such as reported confidence interval values or standard deviations. Additionally, to compare between cases of when win rates or rewards are recorded, we report the *general metric* used.

A5. On occasion, data is only provided from performance plots and not from tables. Hence our dataset records whether data is presented using plots or in tabular form. When data is only provided by plots, we record the final value for a given metric as shown on a plot. For the purpose of our records being as accurate as possible, we ensure these values are within 5% of their true plotted value. Since we cannot exactly determine the confidence bounds from plots alone we omit recording such values in these cases. However, we do still record the type of uncertainty measure used, as presented by the author (and where available elsewhere, the uncertainty values).

A6. The evaluation intervals (evaluation frequency) and independent evaluations per interval (evaluation duration) were also recorded along with their units (e.g. episodes or timesteps). This includes the number of training runs and number of random seeds used. Here, evaluation intervals that refer to the same measurement across papers were standardized (e.g. rounds are changed to episodes).

A7. We record whether reported results are from previous works, i.e. when reported results are from other cited papers and are not reproduced in the particular paper being recorded.

A.3.1 Environment annotations

- All SMAC win rates are reported as percentages (out of 100) and not probabilities (out of 1).
- We record an environment as paper-specific if it is created by the authors of a particular paper and is not utilized in any other article.

A.3.2 Algorithm annotations

In the process of collecting the data for this paper it came to our attention that several algorithms go by slightly different names across multiple papers. For the purposes of our analysis we have standardised these naming choices, based on algorithm descriptions made by authors in their respective papers, to more standardised naming conventions. *IAC-V* is first mentioned in the paper that presents COMA Foerster et al. (2018b). Due to the paper emerging very early into the growth of cooperative MARL naming had not yet been normalised however, *IAC-V* is described as a standard advantage actor-critic (AAC) algorithm using parameter sharing and can instead just be referred to as IAC. *PSMADDPG* Mao et al. (2019) is a variant of MADDPG that makes use of parameter sharing which is the norm in many other publications. Interestingly the original MADDPG paper Lowe et al. (2017b) does not make use of this. *PSMADDPG* can be considered to be MADDPG with a different implementation choice and is grouped with MADDPG as the underlying algorithm is not altered. Both *A3C* and *A2C*

are named in the publications used in this analysis Wang et al. (2020b); Jaques et al. (2019). A2C and A3C refer to the method by which the AAC algorithm is implemented to run using multiple parallel workers with A2C being the synchronous and A3C being the asynchronous variant Mnih et al. (2016). Very early MARL papers referred to independent Q learning simply as Deep Q Network Tampuu et al. (2015b). As MARL developed further it became more important to distinguish between independent and centralised learners and DQN is commonly called IQL. Similarly DDPG can be renamed to IDDPG to distinguish it as an independent learning algorithm. The centralised AAC algorithm is also sometimes called a naive critic. Instead we refer to this method as central-V as this is the first formalised name for this algorithm that we could find Foerster et al. (2016). Finally MAPPO Yu et al. (2021b) is referred to as MAPPO-shared for MAPPO with parameter sharing. However, parameter sharing if the norm amongst most cooperative MARL publications therefore, MAPPO-shared is simply renamed to MAPPO.

Table 2: Algorithm annotations

Name from paper	Standardised naming	Our interpretation
IAC-V (Foerster et al., 2018b)	IAC	IAC-V is the same as IAC.
PSMADDPG (Mao et al. (2019))	MADDPG	The PS denotes parameter sharing.
A3C (Wang et al., 2020b)	IAC	Asynchronous parallelization method for IAC.
A2C (Jaques et al., 2019)	IAC	Synchronous parallelization method for IAC.
MADQN (Tampuu et al., 2015b)	IQL	Old naming conventions.
Naïve critic (Su et al., 2021)	Central-V	Naïve critic is the same as central-v.
MAPPO-shared (Lee et al., 2021)	MAPPO	Parameter sharing is the norm.
MADR (Park et al., 2020)	MADDPG	MADDPG with recurrency.
DDPG (Lowe et al., 2017b)	IDDPG	Denote as independent learner.
DQN (Tampuu et al., 2015b)	IQL	Denote as independent learner.

B Additional Analysis

This section provides additional insights from further analysis on our dataset of performance evaluation for cooperative MARL algorithms.

B.1 Environment

B.1.1 Most used settings

In this section, we are primarily interested in highlighting some of our further findings from the main papers. We first illustrate some of the most widely used settings for the most popular environments as illustrated in Table 3. It should be noted that this analysis was conducted over 29 unique environments with 164 unique scenarios.

Table 3: Most applicable parameters in each environment for the main papers

Environment	Metric	R. Seed	Aggregate Function	Independent variable	Maps/Tasks	Mentions
SMAC	Win Rate (83.3%)	5 (41.7%)	Median (48.4%)	Timestep (97.3%)	39	37
MPE	Reward (40%)	5 (34.8%)	Mean (85%)	Episode (48%)	25	33
Matrix Games	Return (100%)	5-10-100	Mean (100%)	Timestep (98.7%)	-	9
MazeBase	Win Rate (87.5%)	5 (80%)	Mean (100%)	Episode (44.1%)	2	7

StarCraft Multi-Agent Challenge (SMAC): is a partially observable environment, with a diverse set of sophisticated micro-actions that enable the learning of complex interactions amongst collaborating agents, the fundamental concept of SMAC is a team of agents battling against another group of units. SMAC is the most widely used environment in our analysis, since it is employed as the experimental environment in 37 of the main papers presenting 46.9% of the collected evaluation data. This finding is not surprising as we have recorded 39 unique SMAC scenarios with varying scales of difficulty. Moreover, many authors agree that SMAC offers a fair comparison of different algorithms since it provides an open-source Python-based implementation of numerous fundamental MARL algorithms.

Multi-Agent Particle Environment (MPE): is an environment that can be fully or partially observable, cooperative or competitive, and allow communication within some of its tasks. In this environment the agents primarily interact with the landmarks and other entities to achieve various goals. We discover that 33 of the 75 papers employ MPE for algorithm testing, accounting for 20.3% of the collected evaluation data. MPE, like SMAC, is a diversified environment with 25 tasks; nevertheless, we observe a disparity in their utilization, with 27.3% of the main papers utilizing Predator and Prey, followed by Spread which is used in 22.7% of the collected main papers which use the MPE environment.

B.1.2 Evolution of environment usage in MARL

In the early years of MARL research there was a shortage of established multi-agent environments, as shown in figure 8 [9]. Hence most publications tested their algorithms on environments created by the authors (paper-specific environments) as well as MazeBase. Although MazeBase was developed for single-agent environments, it is easily adaptable to the multi-agent case and was used to create the traffic junction combat tasks. This adaptability drove its early adoption. The Figure depicts that, since 2017, we can observe an increase in the use of MPE tasks like Predator-Prey and Spread, as well as StarCraft unit micromanagement. MPE was the most used environment in 2019 and, since 2020, we see SMAC dominating the others.

B.2 Algorithms

B.2.1 Training schemes analysis

Independent Learning (IL or DTDE): is a method that extends single-agent RL algorithms to the multi-agent space. Agents learn an independent policy based on their own local observations and, in the cooperative case, learn a policy based on a shared global reward. This type of learning

[9]The plotted environments occur in at least two papers.

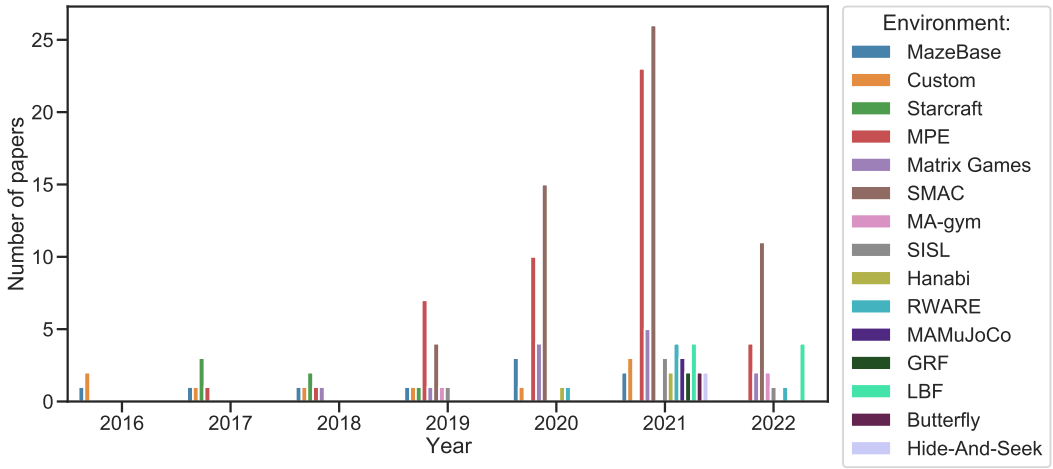

Figure 8: Evolution of environment usage in all the papers

has low convergence guarantees because the learning of other agents causes the environment to appear non-stationary to each individual agent since the agents' behavior changes the dynamics of the environment.

Centralised Training Decentralised Execution (CTDE): much like IL, CTDE learns decentralised agent policies where agents act based on local observations. However, in the CTDE paradigm we can make use of additional information at training time that is normally not available to agents during execution. Typically this is done by using a *centralised-critic* or some *mixing network* which is allowed to condition on the global environment state information or, has access to open communication channels with all agents. The *centralised-critic* or *mixing network* is only used during training time which aids in finding better agent polices during training time without increasing computational overheads during execution time.

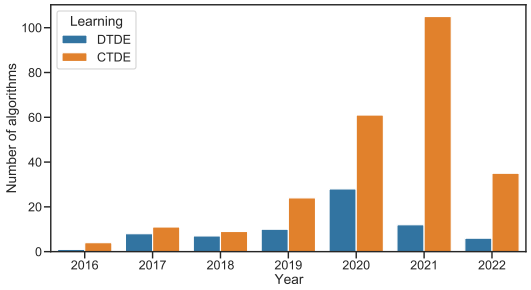

Figure 9: Number of algorithms based on learning schemes by years for all the papers

Is the decline in the use of IL over time a positive or negative sign? CTDE has been demonstrated to be a powerful approach that outperforms decentralized training in many cases. Nevertheless, we cannot assume that it is the optimal solution in all cooperative MARL cases, since many studies, have shown that it is still hard for agents to act cooperatively during execution. This is because partial observability and stochasticity can easily break the learned cooperative strategy, resulting in miscoordination. Recently, we observe the increase of communication algorithms. These can make use of *graph neural networks* as a communication channel to help agents obtain information during both training and execution.

B.2.2 Benchmark algorithms

In our analysis, we examine 150 algorithms where 73.3% are used only once over the 75 main papers. In this section, we provide additional insights from the analysis of our data on the most relevant algorithms. We summarize the use of these algorithms in our dataset in Table 4.

We note that one can select approximately five of these widely used algorithms, from Table 4, as baselines, against which one can evaluate the performance of a novel algorithm. As these algorithms are well-studied they may provide a meaningful current set for comparison. Athough we list these baselines, we do not consider this list to be exhaustive and researchers should strive to compare their

Table 4: Most used algorithms in the main papers

Algorithms	Type of agent	CTDE	Policy	Mentions
QMIX (Rashid et al., 2018b)	Value-based	Yes	Off	35
MADDPG (Lowe et al., 2017b)	Actor-critic	Yes	Off	25
VDN (Sunehag et al., 2018)	Value-based	Yes	Off	23
COMA (Foerster et al., 2018b)	Actor-critic	Yes	On	22
IQL (Tampuu et al., 2015b)	Value-based	No	Off	20
MAPPO (Yu et al., 2021b)	Actor-critic	Yes	On	10
QPLEX (Wang et al., 2021c)	Value-based	Yes	Off	10
QTRAN (Son et al., 2019)	Value-based	Yes	Off	08
IAC (Foerster et al., 2018b)	Actor-critic	No	On	08
CommNet (Sukhbaatar et al., 2016)	Policy optimization	-	-	06

algorithms to algorithms that are currently known to have state of the art (SOTA) performance. The five baselines we choose for discussion encompass both the CTDE and IL paradigm for cooperative MARL as well as policy gradient (PG) and Q-learning based methods. To meet these requirements we discuss QMIX (Rashid et al., 2018b), MADDPG (Lowe et al., 2017b), COMA (Foerster et al., 2018b), IQL (Tampuu et al., 2015b) and MAPPO (Yu et al., 2021b). QMIX is selected as it introduced the concept of monotonic value-decomposition which formed the basis for the development of many of the recent algorithmic developments. As shown by (Hu et al., 2021), fine-tuned implementations of QMIX can still outperform newer methods that attempt to improve upon the original work. We discuss MADPPG since it was introduced in the most widely cited MARL algorithm paper with 2070 citation at the time of writing. We also note that MADDPG provides a baseline for algorithms that are used in mixed and competitive tasks. Although MADDPG was introduced as an algorithm to be used on environments with continuous action spaces, the algorithm may also be adapted to the discrete case. We discuss CommNet since it is a widely used algorithm, used in scenarios which require agent communication in order to find optimal solutions. Furthermore we discuss MAPPO due to recent work illustrating it's effectiveness in cooperative MARL tasks (Yu et al., 2021b). Lastly, we discuss COMA since it is a widely used actor-critic algorithm. Moreover, each of the algorithms mentioned have open-sourced code implementations available (Samvelyan et al., 2019b; Papoudakis et al., 2021; Hu et al., 2021) which serve to decrease the amount of time researchers have to spend on implementing baselines to evaluate against.

QMIX: is a value-based algorithm introduced by (Rashid et al., 2018b) following on from the success of VDN (Sunehag et al., 2018) in cooperative MARL tasks. Similarly to VDN, QMIX makes use of a factorized joint Q-value function to train all agents. What differentiates QMIX from VDN is that individual agents' utilities are joined using a mixing network instead of only summing them. Furthermore, the mixing network is constrained to having only positive weights, leading to a monotonic factorisation of individual agent utilities, and is allowed to condition on the global environment state during training time. QMIX follows the CTDE training paradigm and makes use of recurrent neural networks for individual agent policies. This enables agents to learn joint policies in partially observable settings. The initial performance of QMIX was illustrated by (Rashid et al., 2018b) on the SMAC benchmark.

In our analysis of QMIX, variants of QMIX and algorithms building on QMIX feature most prominently in the 2s3z (18), 3s vs 5z (14), 3s5z (14), MMM2 (13) and 6h vs 8z (11) SMAC scenarios. With numbers in parenthesis denoting the number of papers in which a QMIX variant is benchmarked on a particular scenario.

CommNet: (Sukhbaatar et al., 2016) seeks to address the issue of effective agent communication in partially observable cooperative settings. What differentiated CommNet from previous communication works is that the communication protocol between agents is not fixed, but instead learnt as a neural model alongside agent training. This is possible due to agent communication being modeled using a continuous, differentiable vector which is output by each agent. We find that CommNET is used, most widely, on the TrafficJunction suite of environments which we find to be one of the most widely used communication benchmarks for MARL.

Multi-Agent Deep Deterministic Policy Gradient (MADDPG): introduced by (Lowe et al., 2017b), is a multi-agent extension to the DDPG algorithm introduced by (Lillicrap et al., 2015). MADDPG is an off-policy actor-critic type of algorithm. By default, each agent has a unique policy network and Q-value critic network. Each agent's policy is only allowed to condition on an agent's partial observation of the full environment state while, during training time, each critic conditions on the actions selected by the policy networks of all other agents. MADDPG makes use of standard MLPs for both the agent policy and critic networks but variations of MADDPG exist which make use of recurrent neural networks (RNNs) for agent policies. Similarly, variations of MADDPG exist which make use of weight sharing across agent networks to aid in speedups of algorithm training. An advantage of MADDPG is that the algorithm is inherently applicable to both competitive, cooperative and mixed environments. This versatility is displayed in the seminal paper by (Lowe et al., 2017b).
In our analysis, MADPPG is most widely used for benchmarking on the multi-agent particle environment suite (MPE) with the algorithm being most widely used on the Predator-Prey (12), Spread (10) and Speaker-Listener (5) scenarios.

Multi-Agent Proximal Policy Optimization (MAPPO): is a multi-agent extension to the single-agent Proximal Policy Optimization (PPO) algorithm and mentioned explicitly by (Yu et al., 2021b). Similarly to PPO, MAPPO makes use of a value function, conditioned on the global environment state, to serve as a baseline leading to reduced variance in policy-gradient optimization. Furthermore, MAPPO may be implemented in the CTDE or IL paradigms depending on whether the value function is allowed to condition on some representation of the global environment state or only on an agent's local observation of the environment.
In our analysis, we find that MAPPO is used an equal amount of times (4) on the corridor, (3) MMM2, 5m vs 6m, 3s5z SMAC scenarios as well as on (2) the spread MPE scenario.

Counterfactual Multi-Agent Policy Gradients (COMA): is an actor-critic algorithm the makes use of the CTDE paradigm by using a centralized critic, which is allowed to condition on the full environment state, with decentralized actors. This centralized critic is used during training time only and foregone at execution time. The core contribution of COMA is through addressing the agent credit assignment issue in MARL by utilizing a *counterfactual* advantage function that is unique to each agent. In our analysis we find that COMA is used most frequently in the 2s3z (11), 3s5z (7), 1c3s5z (7) and the 3m (6) SMAC scenarios, as well (6) the Spread scenario from MPE.

B.3 Evaluation Settings

B.3.1 Metric

In general, metrics are used to monitor and quantify a model's performance. Through our analysis, we identify 25 unique metrics, and after unifying the data, based on our annotations as given in A4, we obtain 12 general metrics over the published papers.

The most common three metrics, referred to in our data are, **Return**, **Reward** and **Win Rate** which are in 31.3%, 14.6% and 50% of the main papers respectively. It is interesting to note that **Win Rate** is such a widely used metric, especially since it is environment specific. We believe this high percentage is due to the high use of the SMAC and Traffic junctions environments which commonly use Win Rate.

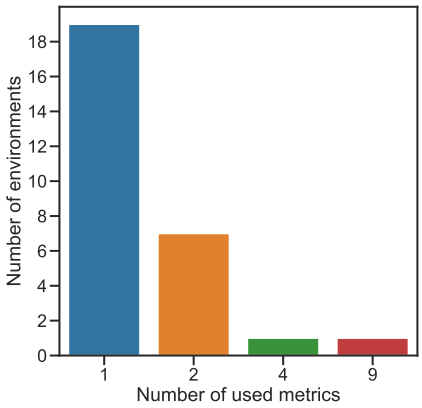

Figure 10: Number of metrics used per environment.

We observe dependencies between the choice of the environment and metrics. Out of the collected SMAC data from the main papers 80.9% use *Win Rate* as a metric, meanwhile in MPE, out of the 720 rows of collected data related to the MPE environment, 35% use *Return* and 25.1% use *Reward*. Moreover, out of 29 environments over the main papers, we find 19 use only one metric type.

B.3.2 Independent runs

Independent training runs can take place across different **random seeds**. In some experiments multiple runs are completed for each random seed, for a fixed set of random seeds. Fixing the random seed is an attempt to control some of the experiment's sources of randomness. The number of runs is important in determining the reliability of the evaluation. More independent runs provide more data which allows for authors to report more accurate measures of spread alongside the point estimates of algorithm performance.
The authors used to employ 10 to 20 runs in the Unit Micromanagement version of the StarCraft environment, but since StarCraft II (SMAC) emerged, authors tend to use only around 5 independent runs. This decline may be due to the environment being more computationally expensive to run. However, we argue, similarly to (Agarwal et al., 2021) for the importance of having 10 independent training runs for reliable confidence intervals.

B.3.3 Aggregate function

An aggregate function, also known as a measure of central tendency, is a single value that intends to portray information about multiple results by determining the central position among a group of various results. For aggregations over algorithm performance, we differentiate between two aggregation steps: the first, which we refer to as the *local aggregate function*, denotes how aggregation is done across evaluation episodes/evaluation runs in a fixed training run. The second, is the *global aggregate function*. This denotes how we aggregate across independent training runs.
The performance of MARL algorithms is often reported using a point estimate of some task performance metric, such as the **mean** and **median** aggregated over the independent training runs. The mean is the most frequently used aggregation function, accounting for over 41.7% of all data gathered from the main papers. It was the only utilized aggregate function in the early years of our recorded dataset. Since 2019, we have seen the introduction of the median as an aggregate function, with the launch of SMAC, and it has became one of the most widely used aggregate functions in SMAC, with some limited use in MPE. The widespread use of the median as an aggregate function can be attributed to the evaluation guideline proposed by (Samvelyan et al., 2019b).

B.3.4 Measure of spread

The measure of spread plays an important role in delivering first hand information about the experiment findings. It expresses how far apart values are in a distribution and it provides a measure of the variability of values obtained across different random seeds or runs. It also serves as a basic way to quantify the uncertainty in a reported point estimate.
In our study, we discovered that 26 out of 75 studies did not mention the measure of spread at all. In some cases this resulted from when performance is only measured over one run, in other cases this is due to a lack of reported details within a paper. In statistics, there are various fundamental measures of spread, the following are the most frequently encountered in our MARL dataset:

Standard deviation: is a common measure of dispersion of a set of values from their mean. The standard deviation will be modest if the values are clustered together. Widely dispersed values will result in a larger standard deviation.

Confidence interval (CI): These provides an estimated possible range for an unknown value. We can choose from a variety of confidence limitations, where some of the most frequent are a 95% or a 99.5% confidence interval.

Inter-quartile range: This is a measure of dispersion which has the advantage of not being impacted by outliers and is important when the researchers want to know where the majority of the findings fall. It is used in 10 papers out of the main ones and it is commonly used in SMAC presenting 41.1% of the SMAC collected data over the published cooperative papers.

B.3.5 Time Measurement

Independent Variable: The training and evaluation time is a vital feature that must be stated by the author for a fair comparison of studies. We identified 9 options to define the independent experiment variable from the main papers, but the most commonly used measure is **time-steps**, which is employed in 39 papers, followed by **episodes**, which accounts for 18 studies. We discover an imbalance where the independent variable being used is strongly related to the environment, with 88.3% of SMAC

collected data using time-steps as an independent variable and 39.2% of MPE collected data using episodes.

Evaluation intervals (evaluation frequency): is generally associated with the SMAC evaluation protocol. It refers to the fixed number of time-steps T, after which training is suspended, to be able to evaluate an algorithm for a fixed number of runs/epsiodes E. During these evaluation runs agents are usually only allowed to act greedily and in a decentralized manner. The test win rate is the percentage of episodes e, in E, for which the agents defeat all enemy units within the time limit. Although this is predominantly employed in SMAC experiments, occurring in 13 out of the 37 main papers that use SMAC, this evaluation approach is also used in the MPE and Level-Based Foraging (LBF) environments, with 6 and 2 papers adopting this methodology in these cases, respectively. The evaluation frequency must ideally be associated with a duration E which we record as the evaluation duration.

Number of independent evaluations per interval (evaluation duration): as the name indicates, this is the amount of evaluations that are performed at each evaluation interval. This detail is required if an evaluation frequency is given, but it may also be provided by itself in the case evaluation is performed of the entire duration of an experiment.

B.4 Evaluation procedure, best practices and guideline

In this section, we summarize, in Table 5, the number of papers that abide by the key practices that are recommended in the main body of this paper. We also show what percentage of the main papers and other papers include each specific practice in their evaluation and reporting protocol.

Table 5: Number and percentage of papers recorded that follow the details of the recommended evaluation guideline.

Evaluation and Implementation details	The main papers			The other papers		
	Yes	**No**	**%**	**Yes**	**No**	**%**
Experiment details						
Evaluate on multiple Environments	38	37	50.7%	18	19	48.6%
Evaluate on multiple Scenarios	65	10	86.7%	36	01	97.3%
Evaluation procedure details						
Report the training time	65	10	86.7%	26	11	70.3%
Report the independent runs	53	22	70.7%	26	11	70.3%
Report the global aggregate function	54	21	72.0%	30	07	81.1%
Report the measure of spread	49	26	65.3%	21	16	56.7%
Report the evaluation interval (evaluation frequency)	20	55	26.7%	09	28	24.3%
Report the number of evaluation runs (evaluation duration)	26	49	34.7%	16	11	43.2%
Use statistical tests	01	74	01.3%	00	37	-
Guideline & Best practices						
Training for 2M timesteps	20	55	26.7%	07	30	18.9%
Train on-policy for 20M and off-policy for 2M timesteps	02	73	02.7%	00	37	-
Use independent evaluation episodes per interval with $E = 32$	04	71	05.3%	01	36	02.7%
Evaluation every 10000 timesteps	04	71	05.3%	03	34	08.1%
Use Mean Return metric	14	61	18.7%	05	32	13.5%
Use Absolute metric	02	73	02.7%	00	37	-
Use 95% CI as a measure of spread	16	59	21.3%	02	35	05.4%
Report plot results	71	04	94.7%	33	04	89.2%
Report tabular results	40	35	53.3%	27	10	73.0%
Ablation study	33	42	44.0%	18	19	48.6%
Same baseline algorithms over all the experiment's environments	54	21	72.0%	27	10	73.0%
Aggregate over the different maps and/or environments	08	67	10,7%	02	35	05.4%
Public repository	36	39	48.0%	13	24	35.1%

B.5 About SMAC

In this section we will raise challenges uncovered in our analysis rather than provide answers. All challenges that are highlighted will be accompanied by all of relevant facts, as found in our dataset, and are from the SMAC benchmark. We are not attempting to criticize the current scenarios or the environment itself, but want to emphasize the need of advocating for the use of SMAC to be standardized such that algorithm designers are limited to specified scenarios when testing their algorithms. The reason for this is to ensure fair comparison between works.

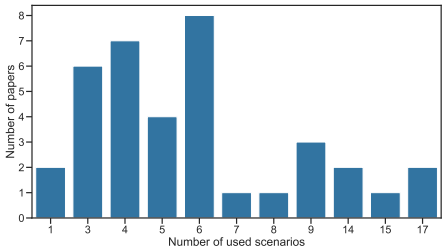

Figure 11: Number of used SMAC scenarios per paper

As we indicated in the environment section, SMAC is a popular benchmarking environment and we discovered that 13 publications out of 75 apply only SMAC to prove the trustworthiness of their experiment. We notice that, despite the fact that SMAC provides many testing scenarios (39 used ones in the published papers), most publications only employ a few of them in their reported trials, as seen in figure 11.

What are the features needed to define the difficulty of a scenario? After analyzing the win rate distribution under various settings, we discovered that several scenarios that were thought to be difficult turned out to be simple through using independent learning algorithms. A clear example of this is illustrated in Figure 12 by the shift in the win rate distribution for CTDE and DTDE algorithms evaluated on the corridor scenario.

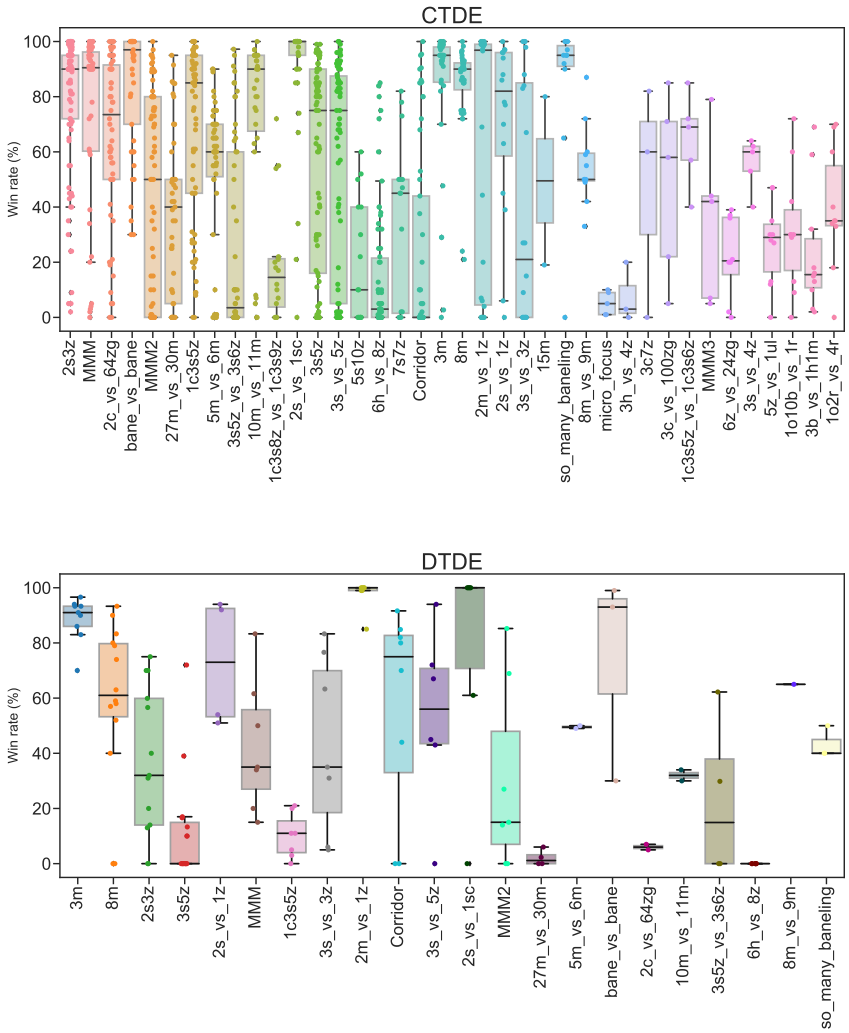

Figure 12: SMAC win rate distribution based on training schemes from the main papers

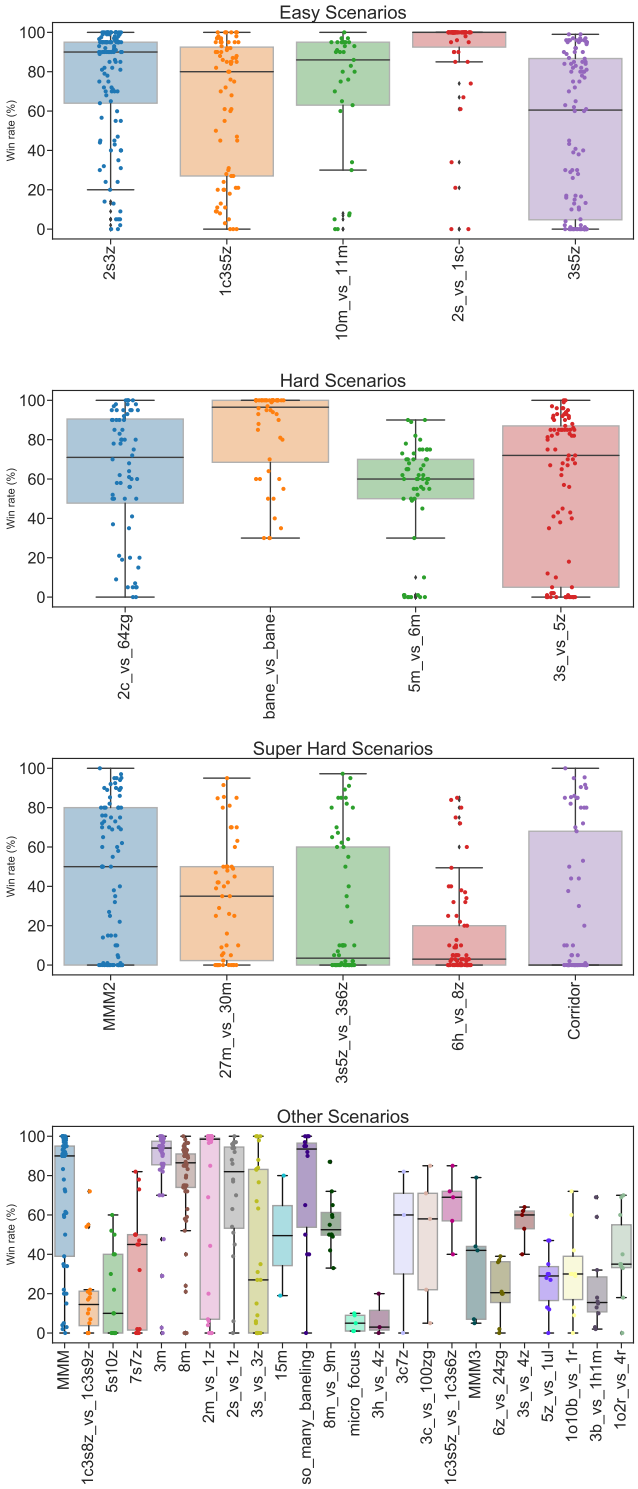

Figure 13: SMAC win rate distribution based on difficulty from the main papers

Furthermore, figure 14 emphasizes the importance of training until 2M timesteps. It demonstrates how the win rate, for even the easiest scenarios, has a wide spread when algorithms are trained for less than 2 million time steps. It can also be noted that, when algorithms are trained up to 2 million timesteps or more, that performance convergences to a higher win rates, not only for easy scenarios but also for hard and even super hard ones.

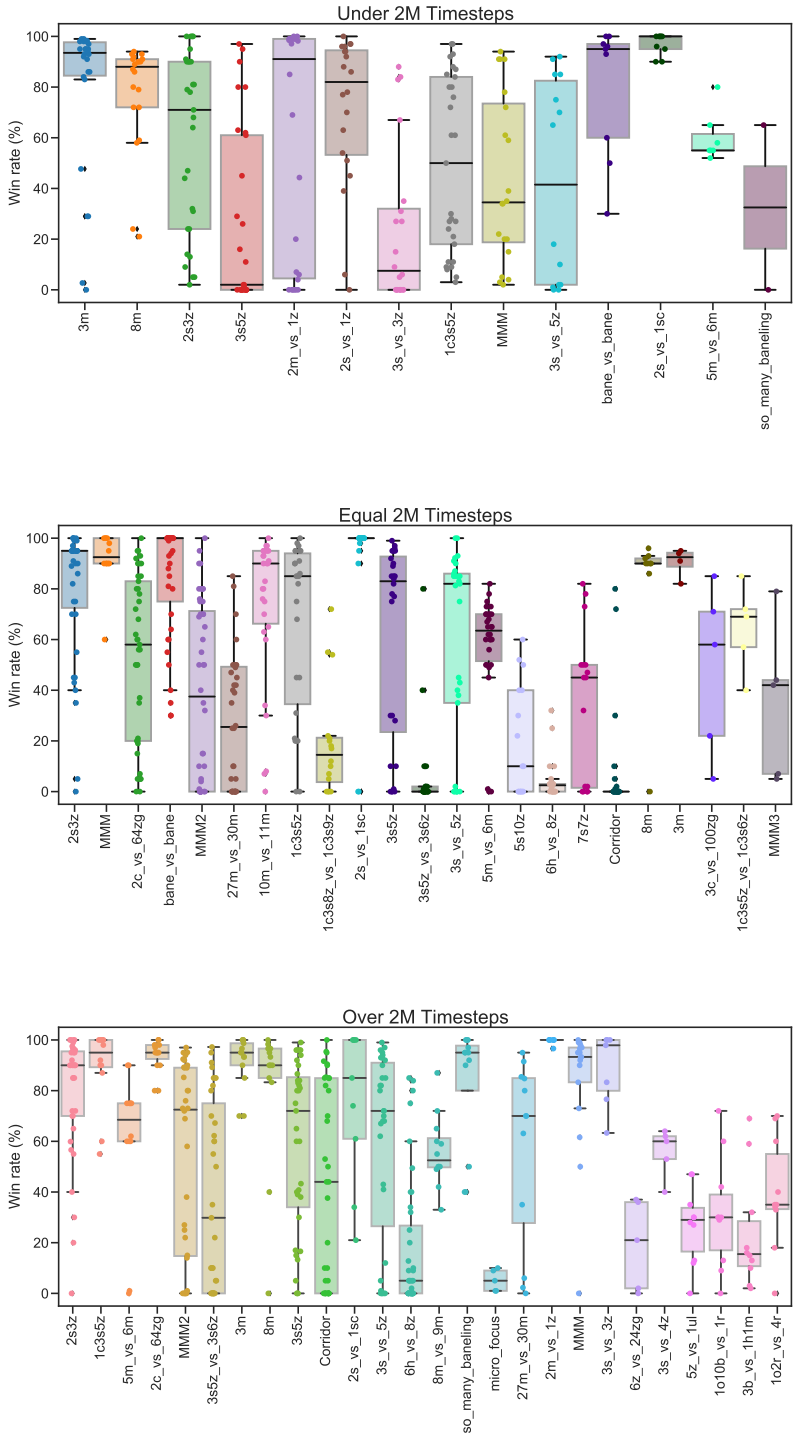

Figure 14: SMAC win rate distribution based on training time from The main papers

Which scenario to choose? The choice of the scenarios for an algorithm designer is a critical task, considering the fact that each scenarios itself in SMAC has its own challenges, which can work in the algorithm's favor (e.g. IA2C in Corridor) or in its misfortune (e.g. IA2C in MMM2). Moreover, 50% of the scenarios were used in one or two papers only, some of these scenarios were used for ablation studies or for a specific research direction like communication, nevertheless most of them do not have prior justification.

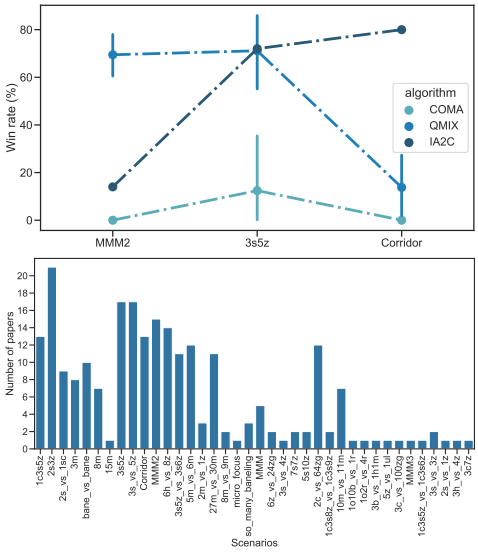

Figure 15: **Top:** The performance of COMA, QMIX and IA2C in 3 different SMAC scenarios. **Bottom:** Number of papers that use each scenario over the main papers

Is the inconsistency in performance inescapable? In Figure 16, we fixed the training steps to be 2 million for all recorded papers that use the bane vs bane, MMM2, 3m and 27m vs 30m SMAC scenarios. We achieve this by reading algorithm performance from plots produced in all relevant papers. It is known that the version of SMAC that is used can have an effect on algorithm performance, but here we see that merely fixing the training time steps across multiple papers leads to even greater performance discrepancies between papers than the SMAC version being used.

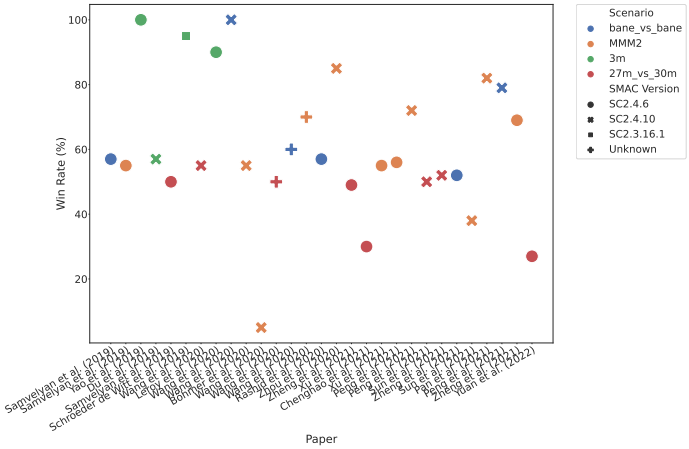

Figure 16: Performance of QMIX on different SMAC scenarios trained for 2M timesteps

C Guideline

C.1 Motivation

In the following section, we will demonstrate our evaluation guideline. We follow the steps as outlined but omit ablation studies in our work since we are not trying to introduce a novel algorithm. We also have not tuned any hyperparameters for any of the algorithms we consider. We wish to make the reader purposefully aware that the primary goal of this experiment is to provide an illustration of how to use our evaluation guideline and our experiment is not focused on the performance of the chosen algorithms. As such we are not striving to achieve state of the art performance on the flatland benchmark and will merely illustrate how results may be interpreted by a researcher.

Please note that, following our guideline, we make all raw data of our experiments available. Our code will be made publicly available soon. We provide a LaTeX template for the proposed reporting templates 6. We envision that such a template will make it easier for other authors to define and report all details pertaining to their experiments. These include experimental details, evaluation protocols, environment settings and all other details that authors wish to report. Of course, it's up to the author to choose the set of hyperparameters to be reported according to the algorithm class and its specific hyperparameters.

C.2 Reporting templates

Here, we present an example of a template that can be used to summarise the important information required to perform the evaluation of algorithms, see Table 6.

Firstly, we have to list all the algorithms we are comparing. For hyperparameters we suggest listing all tunable parameters which are manually set by the researcher. Some parameters like the *discount factor* are fairly consistent throughout published works and can easily be reused across papers. Other parameters can vary through papers due to computational constraints like *batch size* which can be limited by the available memory of the GPU used in training and sometimes needs to be adjusted based on computational limitations and, *replay buffer size* which is limited by the available memory of the training computers. Parameters like the *target network update period* and ϵ *schedule* are required to replicate the training scheme used by an algorithm as when mistuned they greatly alter results Rashid et al. (2021).

Network architecture is also an important consideration for MARL algorithms. QMIX is one of the most popular value-decomposition methods in cooperative MARL and makes use of *hypernetworks* to train the central value function. The parameterisation of these *hypernetworks* must also be noted as their configurations have a strong effect on the effectiveness of the value function. The central value function for QMIX inspired value-decomposition methods is commonly called a *mixing network* and is responsible for performing multi-agent credit assignment during training. The *mixing networks* can vary across different methods but, without knowing how they are parameterised it is possible for networks to have a large variance in their complexity which makes direct comparisons difficult to interpret Hu et al. (2021)

Additionally not all methods consistently make use of recurrency in their architecture which is important for achieving high performance in partially observable settings. Parameter sharing is also unique the cooperative MARL setting and is used in most publications, however, not all papers make use of this paradigm.

Code-level optimisations consist of any parameters that can be included in algorithm implementations for improving performance, but are not core components of the algorithm. *Reward normalisation* is when the rewards over the episode are normalised which reduces variance and makes learning easier Yu et al. (2021b). Not all settings make use of normalised rewards but they can be trivially implemented at the code level. *Death masking* is important to note as different frameworks deal with dead agents in different manners which can make direct comparisons difficult. *Clipped updates* are used in some papers to prevent exploding gradients and can be trivially implemented in most deep learning frameworks. *Eligibility traces* can be used to adjust the variance and bias trade off for return calculations and are tuned using the λ parameter. Although using $TD(\lambda)$ returns has been shown to improve performance for MARL algorithms it is not universally used and must be taken into account for evaluation. Optimiser choice has also been shown to have a large impact on the performance of MARL algorithms and cannot be interchanged arbitrarily (Yu et al. (2021b)).

Computational resources, although not important for algorithmic development are still relevant to research. Clarity of the resources required for a publication to be replicated provide an indication to researchers as how feasible replication is and, how similarly optimised their own implementations are. It also makes it clear where methods may perform better at the cost of compute.

Evaluation protocols need to be made clear in publications so that the results are easy to interpret. By providing all evaluation in the template details readers do not need to pick through a paper to determine how to interpret results. The evaluation framework and the version that is being used is also of importance. Evaluation frameworks are frequently updated and results might be incomparable in-between versions.

Finally it is important to provide the configurations of the environments being used to train and evaluate the algorithm. On one hand, in sample evaluation allows to evaluate an algorithm's performance on an environment configuration similar to the configurations it was trained on. On the other hand, out-of sample configurations help to test the ability of the algorithm to generalise to a different configuration of the environment that was not seen in the training. It is obvious that there are many standardised settings in MARL. There are also cases of publications using custom environments which are non-standard when compared to existing publications. These non-standard settings require a full detailing of specifications to make them easier to understand.

Table 6: Proposal for reporting experimental details

Experimental setup	**Algo 1**	**Algo 2**	**Algo 3**
Hyperparameters			
Discount factor			
Batch size			
Replay buffer size			
Minimum replay buffer size before updating			
N steps bootstrapping			
Target network update period			
ϵ schedule (Decay steps, ϵ start, ϵ min)			
Value Network architecture			
Value Network initializer			
Value Network Layer size			
Value Network Layer normalisation			
Mixing network (architecture, size, activation)			
Hypernetworks (size, activation)			
Parameter sharing			
Parallel workers			
Seed range			
Code-level optimisations			
Optimiser (type, parameters)			
Learning rate			
Reward normalisation			
Death masking			
Clipped updates			
Eligibility trace			
$TD(\lambda)$ value			
Computational resources			
Average Wall-clock time per algorithm			
CPUs per experiment			
GPU per experiment			
RAM per experiment			
Evaluation protocol			
Total training (timesteps)			
Evaluation interval (timesteps)			
Independent evaluation episodes			
Absolute metric (evaluation episodes, aggregation method)			
Local aggregation method			
Global aggregation method			
Metrics [Environment 1 name]			
Metrics [Environment 2 name]			
Metrics [Environment 3 name]			
Exploration behaviour			
MARL Framework		name (version)	
Environment settings			
Environment 1 name (version)	**Training**	**In sample evaluation configs**	**Out of sample evaluation configs**
Env related configs			
Environment 2 name (version)	**Training**	**In sample evaluation configs**	**Out of sample evaluation configs**
Env related configs			
Environment 3 name (version)	**Training**	**In sample evaluation configs**	**Out of sample evaluation configs**
Env related configs			

C.3 Experiment details

Firstly we note the algorithms used for the experiments. For illustration purposes we use IQL which is an independent learning algorithm, VDN which is a linear value-decomposition method and finally QMIX which is a value-decomposition method that makes use of a central value function. It is important to note that not all parameters are applicable to all types of algorithms.

C.3.1 Environment

An environment can present various factors of variations forming two different context sets: the first being the set of all supported random seeds which makes use of Procedural Content Generation (PCG) and the second is the product of multiple factors of variations inside the environment. It has been noted that procedurally generated environments may reduce the precision of research Kirk et al. (2021) while being able to control a factor of variations in an environment offers more flexibility to create environment configurations that match the evaluation of different algorithmic strengths. Regardless of the context being used, we strongly advocate that researchers should report all the environment settings used for training and for evaluation, see the environment settings section in Table 6, as an example of reporting environment settings. Of course, all settings are environment specific.

For our experiments, we make use of the Flatland benchmark environment Mohanty et al. (2020) first introduced as a challenge in 2020 to investigate solutions to the vehicle rescheduling problem in railway systems. At a high-level, Flatland is a highly customisable, simplified 2D grid environment which aims to simulate the routing of trains from one city to another.

Figure 17: Flatland maps varying between consecutive environment episodes.

We particularly choose Flatland as a benchmarking environment due to the fact that the environment may be set to be non-static allowing it to change after each completed episode during both training and evaluation. This enables us to test the ability of algorithms to generalise outside of experience that was encountered during training. Flatland offers a high level of customisability with regards to this environment regeneration, but we opt for a relatively limited and simple approach. Once a map has been generated, we keep the rails of the map fixed but we allow the number of stations on the map to be randomly distributed at each new episode. This changes the location on the map where an agent starts at each episode as well as the destination that each agent must reach. An example of how the maps might change over 3 episodes is demonstrated in Figure 17.

Observation space. For the observations of each agent, we make use of what the Flatland authors refer to as tree observations. For these observations, an agent is allowed to construct a tree in four directions which follows permitted transitions. These trees are allowed to pass a fixed number of points on the grid where more than one action is allowed with these points being referred to as *switches* by the authors. Each agent is then allowed to observe the grid up to a fixed number of switches (referred to as the maximum permitted tree depth) and then constructs local features based on the observed tree. These features then inform the agent's decision making.

For all algorithms, each agent only makes use of its own local tree observation to inform its action selection. Since Flatland does not return a global state for the entire grid at each training time step, on which QMIX can condition its mixing network during training time, we construct a simple global state representation which is the concatenation of all agents' local observations.

Action space. The action space in flatland is $discrete(5)$ and consists of the following actions:

- Move forward,
- Select a left turn,
- Select a right turn,

- Halt on the current cell,
- Take no action.

Reward structure. At each environment time step each agent, i receives a reward calculated as:

$$r_i(t) = \alpha r_l^i(t) + \beta r_g(t)$$

Here r_l^i denotes an agent's local reward which is -1 for all time steps until an agent reaches its destination after which it is 0 until episode termination. r_g denotes an additional team reward of 1 which is received by all agents when all agents have reached their destination during an episode. α and β are adjustable parameters which govern agent cooperation.

After an episode is completed each agent receives a return g_i which is computed as:

$$g_i = \sum_{t=1}^{T} r_i(t)$$

In order to keep track of team performance, we monitor and report the mean team episode return which may be calculated for N agents as

$$g_t = \frac{1}{N} \sum_{i}^{N} g_i$$

Aside from only keeping track of the team return, we also record the team completion rate which is the proportion of agents that were able to reach their destination in a given episode.

C.4 Evaluation protocol and experimental procedure

Detailed flatland experiment settings are given in Table 7. We perform 10 independent runs, each with a unique random seed for the initialization of the agent policy networks. For each independent run we evaluate algorithm performance for 32 episodes at every 10000 environment time steps. During these evaluation intervals we freeze training such that agent policy network weights remain fixed and agents are only allowed to act greedily by selecting actions which an agent believes to have to highest Q-values. In order to report the overall team performance, we report the mean return and completion over of all agents in the environment at each episode. It should be noted that in Flatland agents receive their reward at the end of an episode and therefore episode returns and rewards are equivalent. In order to normalize the episode returns we keep track of the maximum and minimum return obtained over all evaluation episodes done during training for a given independent run and then normalize the mean episode return of each evaluation episode according to these global maximum and minimum values. In order to obtain the per task results, we compute the mean and 95% CI over all independent runs at each evaluation interval for both the normalised mean returns and the completion rate. Additionally, for each independent training run, we keep track of both the maximum mean return and maximum completion rate computed at each evaluation interval and use these values to checkpoint the agent network parameters where performance for both these metrics are optimal. Once an independent run is complete we then evaluate the algorithm greedily for 320 episodes using the best model parameters found for both the mean episode return and completion rate and take the mean over these roll outs to compute the absolute metrics for both the completion rate and the mean episode return. In all cases we opt to use the mean instead of the inter-quartile mean since we assume there to be relatively few outliers due to the fact that all results are generated using the same fixed policy. For each independent run, we then normalise the absolute metrics across all algorithms that were being tested such that all absolute metrics fall within the range $[0, 1]$. In all cases it should be noted that, since the goal of normalisation is to constrain metrics to lie within the same $[0, 1]$ interval we omit normalising metrics that inherently lie on such a range, like the completion rate. Since we have only one task, we then construct a (10×1) vector per algorithm using the obtained normalised metrics in order to make use of the tools provided by (Agarwal et al., 2021) and obtain the following results.

Table 7: Reporting Flatland experimental details

Experimental setup	**IQL**	**QMIX**	**VDN**
Hyperparameters			
Discount factor	0.99	0.99	0.99
Batch size	32	32	32
Replay buffer size	5000	5000	5000
Minimum replay buffer size before updating	32	32	32
N steps bootstrapping	5	5	5
Target network update period	100	200	200
ϵ schedule (Decay steps, ϵ start, ϵ min)	(100000,1.0,0.05)	(100000,1.0,0.05)	(100000,1.0,0.05)
Value Network architecture	Recurrent	Recurrent	Recurrent
Value Network initializer	Variance Scaling	Variance Scaling	Variance Scaling
Value Network Layer size	[64,64] GRU	[64,64] GRU	[64,64] GRU
Value Network Layer normalisation	True	True	True
Mixing network (architecture, size, activation)	-	Feedforward,[32], ReLU	-
Hypernetworks (size, activation)	-	[64], ReLU	-
Parameter sharing	Yes	Yes	Yes
Parallel workers	8	8	8
Seed range	{0..9}	{0..9}	{0..9}
Code-level optimisations			
Optimiser (type, parameters)	Adam	Adam	Adam
Learning rate	1e-4	1e-4	1e-4
Computational resources			
Average Wall-clock time per algorithm	9h27m	9h36m	9h16m
CPUs per experiment		20	
GPU per experiment		1	
RAM per experiment		20 GB	
Evaluation protocol			
Total training (timesteps)		2000000	
Evaluation interval (timesteps)		10000	
Independent evaluation episodes		32	
Absolute metric (evaluation episodes, aggregation method)		320, Mean with normal 95% CI	
Local aggregation method		Mean	
Global aggregation method		IQM with 95% stratified bootstrap CI	
Metrics [Flatland]		Return, Completion rate, Normalised score	
Exploration behaviour		Disabled	
MARL Framework		MAVA (0.1.2)	
Environment settings			
Flatland (3.0.15)	**Training** [10]	**In sample evaluation**	
Number of agents	5	5	
Grid size (width x height)	25x25	25x25	
Maximum number of cities	4	4	
Maximum rails between cities	2	2	
Maximum rails in city	3	3	
Malfunctioning rate	0	0	
Observation (type, depth)	TreeObservation, 2	TreeObservation, 2	
Shortest Path Predictor max depth	30	30	
Grid mode	True	True	
Regenerate schedule on reset	True	True	
Regenerate rail on reset	True	True	
Seed	0	0	

C.5 Results

All plots that are generated here are made using the tools provided by (Agarwal et al., 2021).

C.5.1 Sample efficiency curves

The sample efficiency curves serve as a way to asses an algorithm's experience efficiency at improving on a particular metric during training time. For two algorithms that achieve the same final performance on some metric, the algorithm that does so with less environment steps could therefore be considered to be more sample efficient. We compute the sample efficiency curves by making use of the normalized mean return at each evaluation interval as well as the mean completion rate achieved at each evaluation interval.

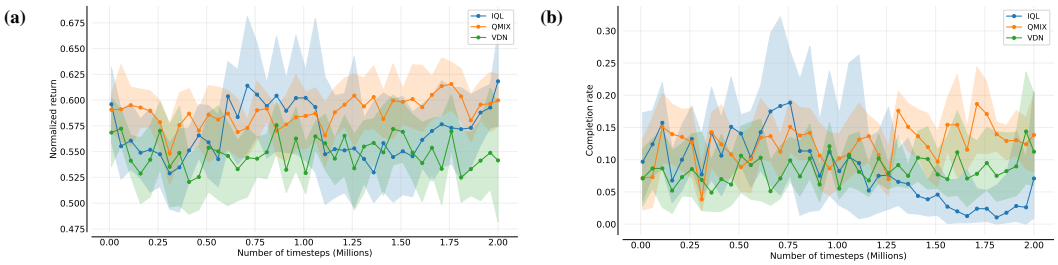

Figure 18: *Sample efficiency curves for experiments*. **(a)** Normalized return. **(b)** Completion rate.

From Figure 18 it can be noted that no particular algorithm is more efficient than any other algorithm and that all algorithms achieve relatively similar final performance. From Figure 18 (a) it can be noted that IQL and QMIX do reach a slightly higher final mean return than VDN.

C.5.2 Aggregate score performance

All aggregated scores are done using the aggregation functions as shown in Figure 19. One aggregation function to note is the *Optimality Gap* which may be thought of as the how far an algorithm is from optimal performance at a given task. For this reason, a lower score is considered to be desirable. The confidence intervals shown alongside the point estimates (black bars) are the 95% stratified bootstrap confidence intervals.

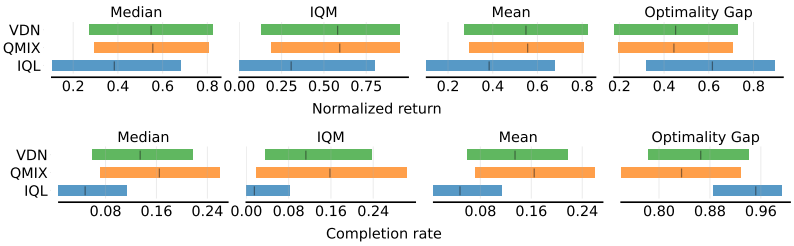

Figure 19: *Per task performance on a* 25×25 *flatland grid*. **(Top)** Normalized return. **(Bottom)** Completion rate.

One can note from the top and bottom figure in Figure 19 that there is large variance in algorithm performance for both metrics used and it is hard to distinguish which algorithm has superior performance, particularly between VDN and QMIX. A clear outlier is IQL which consistently performs worse, across all metrics, than VDN and QMIX.

[10] Using same generator config from https://gitlab.aicrowd.com/flatland/neurips2020-flatland-baselines/-/blob/flatland-paper-baselines/envs/flatland/generator_configs/small_v0.yaml

C.5.3 Performance profiles

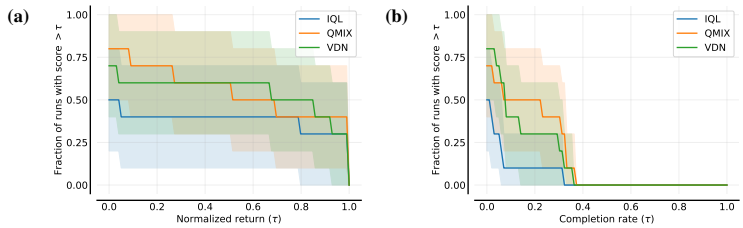

Figure 20: *Performance profiles for experiments*. **(a)** Normalized return. **(b)** Completion rate.

From Figure 20 in can be noted that the performance profiles paint a similar picture to the sample efficiency curves and the aggregated algorithm scores. IQL is consistently outperformed by VDN and QMIX. The performance profiles also clearly illustrate the no algorithm achieves a particularly high completion rate, highlighting the poor overall performance of all algorithms on the environment task.

C.5.4 Probability of improvement

Probability of improvement plots should be interpreted as the probability that an algorithm X has superior performance than algorithm Y with a low score indicating that algorithm Y likely to be better than algorithm X and vice versa for a high score.

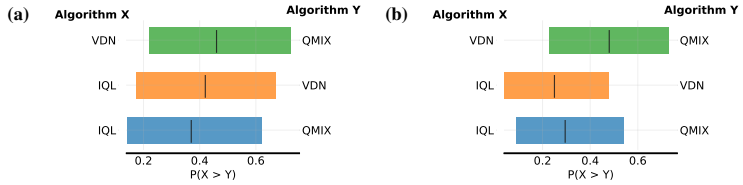

Figure 21: *Probability of improvement plots*. **(a)** Normalized return. **(b)** Completion rate.

From Figure 21 one can note again that IQL is outperformed by VDN and QMIX using both metrics considered. It can also be noted again, that the performance of VDN and QMIX are relatively similar.

C.5.5 Tabular Results

We report the IQM aggregated over the task for all algorithms with the 95% stratified bootstrap CI as well as the mean absolute performance of all algorithms on the task with the 95% CI. These scores collectively, as well as the sample efficiency curves sketch a full picture of the performance for a given algorithm. One can, at a glance, see the performance that the the best policy for a particular algorithm is able to achieve from the tabular results, but one can also get a clear sense of the robustness of a particular algorithm by considering the sample efficiency curves. This makes for transparent result reporting.

The tabular results once again confirm all previous results in that IQL has inferior performance on the task when compared to its value factorisation counterparts and that VDN and QMIX obtain very similar performance. Due to the larger confidence intervals however, no clear conclusions can be drawn since the performance of all algorithms overlap when taking the CIs into account. One can also notice from Tables 8 & 9 that the absolute metric and IQM scores are very similar. The reason from this is because we only consider a single task in our environment. The true power of the tools that were used will be better illustrated when multiple tasks are considered.

C.5.6 Overall findings

Due to the large variance in algorithm performance, we cannot draw any strong conclusions regarding algorithm performance from these experiments, but we have been able to illustrate to use of our guideline and how it gives a full overview of both the absolute and overall performance of a set

Table 8: IQM of absolute metrics for experiments with 95% Stratified Bootstrap CIs

Algorithm	Normalized Returns	Completion Rate
IQL	0.307 (0.0, 0.799)	0.015 (0.0, 0.083)
QMIX	0.593 (0.189, 0.949)	0.158 (0.019, 0.304)
VDN	0.581 (0.131, 0.949	0.113 (0.035, 0.236)

Table 9: Mean per task absolute metrics with 95% CIs

Algorithm	Normalized Returns	Completion Rate
IQL	0.384 (0.08, 0.688)	0.048 (0.00, 0.109)
QMIX	0.556 (0.284, 0.828)	0.164 (0.066, 0.263)
VDN	0.548 (0.254, 0.842)	0.134 (0.05, 0.218)

of algorithms on a particular task. We will continually update our demonstration by adding more flatland tasks, tuning algorithms and, ultimately, adding more environments to this experiment.