Towards a Standardised Performance Evaluation Protocol for Cooperative MARL

Rihab Gorsane [1] Omayma Mahjoub [1] Ruan de Kock [1] Roland Dubb [3]
Siddarth Singh [1] Arnu Pretorius [1] Wiem Khlifi [1,3] Abidine Vall [1,4]

[1]InstaDeep Ltd. [2]National School of Computer Science, Tunisia [3]University of Cape Town, South Africa [4]National School of Engineering of Tunis, Tunisia

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 104 papers accepted for publication from 2016 to 2022, we bring to light worrying trends that put into question the true rate of progress.

From RL to MARL: Lessons, trends and recommendations

1 Know the true source of improvement and report everything

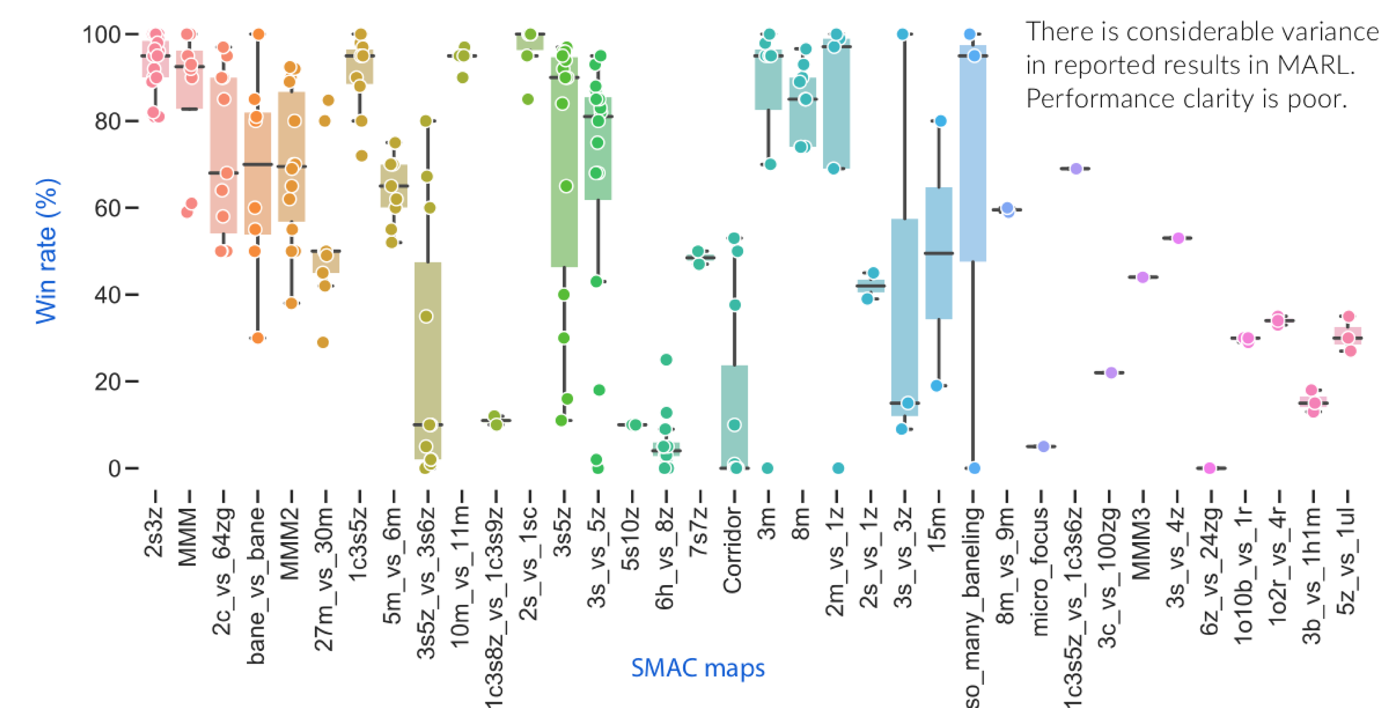

Figure 1. Historical performance of QMIX on different SMAC maps across papers.

2 Use standardised statistical tooling for estimating and reporting uncertainty

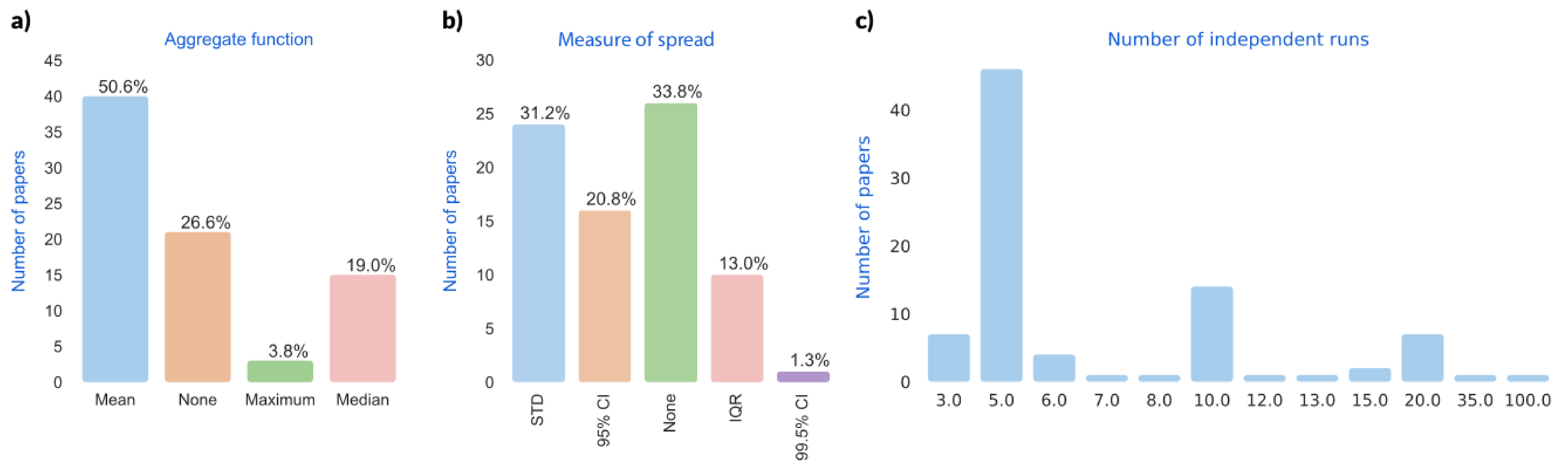

Figure 2. Statistical tools used across papers: a) Aggregate functions, b) Measure of spread c) Independent runs.

3 Guard against environment misuse and overfitting

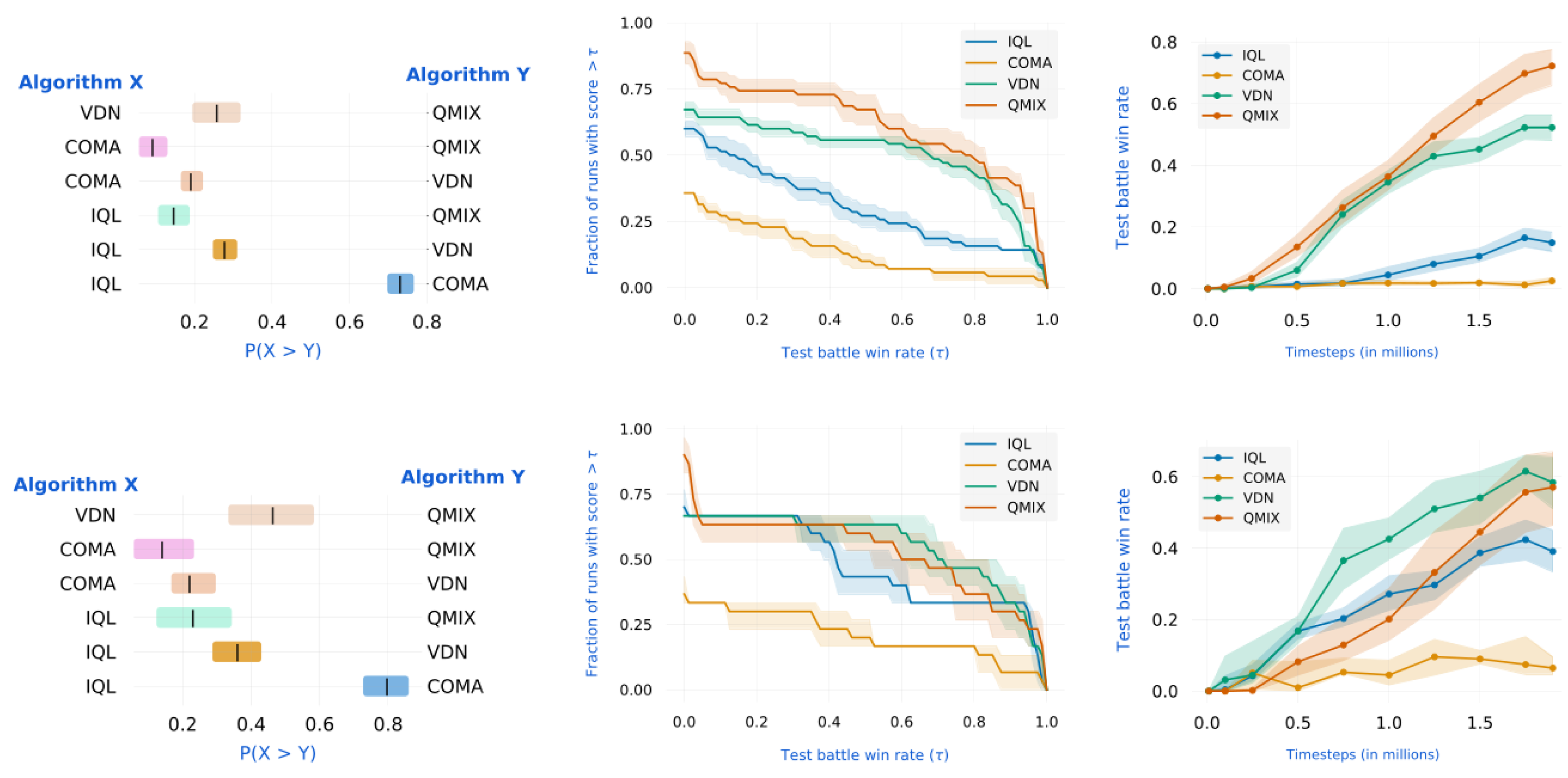

Figure 3. Reanalysis of original SMAC experiments [1]

Contributions

1 We have open sourced our dataset and call for contributions from the MARL community as a whole

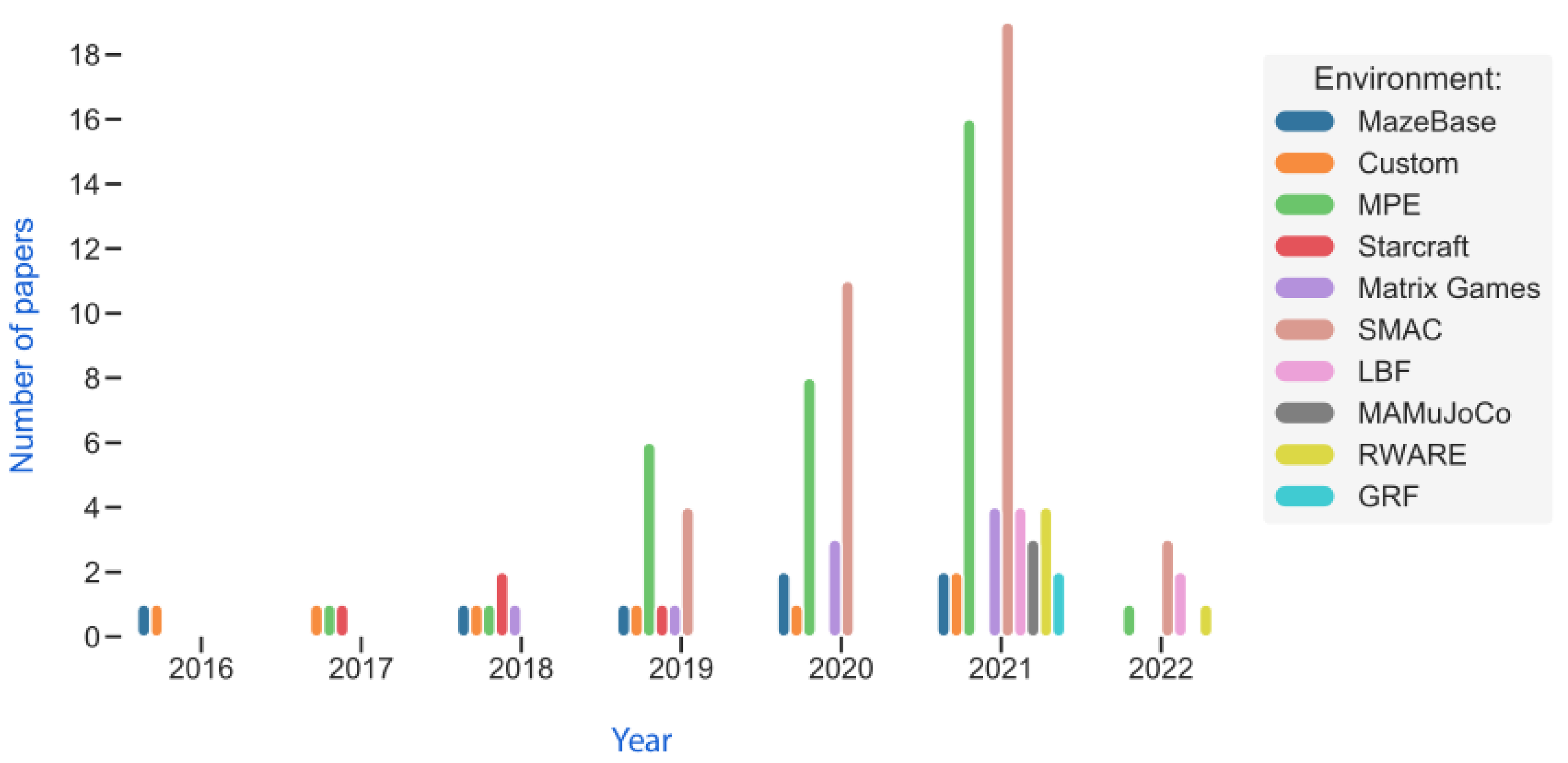

Figure 4. Environment adoption over time.

2 Evaluation Tools Repo

We have open-sourced a repo to do all statistical aggregation and produce all plots given raw experiment data in the correct format.

3 Standardised evaluation protocol

A Standardized 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.
- Number of independent training runs, $R = 10$ (from Agarwal et al. (2022) [1]).
- 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)[2] 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 $(\mathcal{R}, |\mathcal{T}|)$ where each entry is the normalised absolute return for a specific training run on a specific task. 100 Compute the IQM and optimality gap with 95% stratified Bootstrap CIs, probability of improvement scores.
   - 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)[1a] 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 Experimental reporting template

Experimental setup	Algo 1	Algo 2	Algo 3
Hyperparametres			
Discount factor Batch size Replay buffer size Minimum replay buffer size before updating N steps bootstrapping Target network update period € schedule (Decay steps, € start, e min) Value Network architecture Value Network initializer Value Network Layer size Value Network Layer normalisation Mixing network (architecture, size, activation)			

MARL Framework		name (version)	
Environment settings			
Environment 1 name (version)	**Training**	**In sample evalution configs**	**Out of sample evaluation configs**
Env related configs			
Environment 2 name (version)	**Training**	**In sample evalution configs**	**Out of sample evaluation configs**
Env related configs			
Environment 3 name (version)	**Training**	**In sample evalution configs**	**Out of sample evaluation configs**

Additional Findings

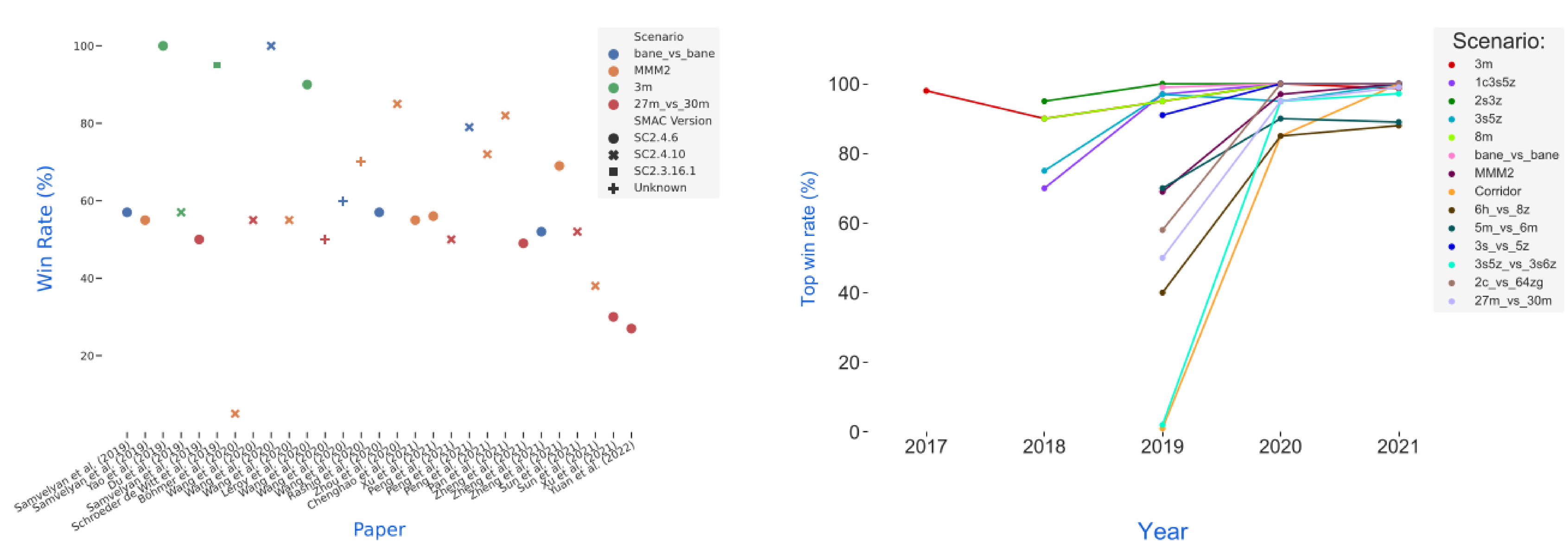

Figure 5. Caption

Conclusion

We propose a 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. It is our hope that, if widely adopted, such a protocol could make comparisons across different works much faster, easier and more accurate leading to sustained progress in the field.

References

[1] 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.

[2] 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.

r.gorsane@instadeep.com