# RelaxedQMIX: Relaxing Monotonicity Constraints for Cooperative Multi-Agent Learning in StarCraft II

**Mi-Ru Youn**

## 1 Introduction

Cooperative multi-agent reinforcement learning (MARL) plays an important role in systems where multiple agents must interact with each other to achieve a common shared objective. Applications of this concept can range from autonomous vehicles to distributed robotics, and even real-time strategy (RTS) games. However, it is important to note that there are several challenges that arise during the implementation of such systems. Partial observability limits the information available to each agent forcing agents to base their decisions on local surroundings. Coordination between agents can also become increasingly difficult as the number of agents grows, exponentially increasing the complexity of interactions. Properly assigning credit to individual actions within a team setting remains a complex problem.

With methods like QMIX, a well-known centralized training with decentralized execution (Oliehoek et al., 2008; Kraemer & Banerjee, 2016) framework, strong performance has been achieved in cooperative tasks. However, this success does not come without limitations. QMIX (Rashid et. al, 2018) uses a mixing network that combines individual agent Q-values into a global Q-value that ensures centralized training and decentralized execution is followed. This approach, however, imposes a strict monotonicity constraint between the individual and total Q-values. Although this provides consistent training and execution, it also limits the variety of cooperative actions agents may learn. In particular, agents are discouraged from taking short-term sacrifices, such as positioning sacrifices or tactical retreats, that could improve the team's long-term success.

In this paper, a simple modification called RelaxedQMIX is introduced. This modification to QMIX aims to relax the strict monotonicity constraint by appending a controlled, learned correction term to the original QMIX output. RelaxedQMIX provides agents with greater flexibility to explore and adopt cooperative strategies that would otherwise be penalized. The goal is to maintain the advantages of centralized training and decentralized execution while enabling agents to adopt smarter, more coordinated behaviors that traditional monotonic architectures typically cannot capture.

The environment on which RelaxedQMIX will be tested is StarCraft II (SC2). SC2 is a real-time strategy (RTS) computer game that provides fast-paced and highly dynamic situations where many agents must cooperate to secure objectives and execute tactical maneuvers. The game is set in a complex and partially observable environment, making it the ideal platform for research in multi-agent reinforcement learning (MARL).

## 2 Background

MARL can be generally divided into two general classes: cooperative and competitive settings.

In competitive environments, agents pursue conflicting objectives, often requiring adversarial strategies that maximize one's reward at the expense of others. In contrast, cooperative multi-agent environments present different challenges, as all agents must coordinate toward a common objective while operating under partial information (i.e., agents don't have access to the full state of the environment; Samvelyan et al. 2019). Issues such as partial observability and limited communication require agents to make decisions based solely on local information and without access to the full global state. To address these difficulties, a framework called centralized training with decentralized execution (Oliehoek et al., 2008; Kraemer & Banerjee, 2016) was introduced, where agents are trained with access to the full global state but must act independently during execution.

**Prior Arts** Over the years, many algorithms have been developed that have either driven advancements or achieved state-of-the-art performance. One of the earliest studies on multi-agent reinforcement learning was conducted by Tan (1993). The primary focus of this study was to compare the difference between independent learning and cooperative learning in multi-agent environments. In independent learning, each agent treats other agents as part of the environment, which leads to challenges such as non-stationarity and slow convergence. As agents continually attempt to optimize their policies, the environment becomes unstable because each agent operates under the assumption that the environment is stationary, even as other agents are simultaneously updating their own strategies. Without proper communication and coordination, agents must rely solely on random exploration to align behavior, which leads to slow convergence. In light of these challenges, independent agents can still learn effective policies given sufficient time. In cooperative learning, agents can share information such as sensory data, episodes, or learned policies, which can significantly accelerate learning and perform better in tasks that require active coordination. However, a trade-off of cooperation is that it expands the effective state space and can exponentially increase communication costs. Tan's work provided early empirical evidence that independent learning struggles particularly in joint tasks and called for greater attention to challenges such as credit assignment and environment non-stationarity in multi-agent systems.

Czarnecki et al. (2017) introduced Value Decomposition Networks (VDN) to address challenges involving partial observability and add more stability to centralized training. At its core, VDN approximates the joint action-value function by summing up every individual Q-value involved. This additive decomposition ensures that decentralized execution is possible while following centralized training. However, because of its inherent simplicity and stability, VDN lacks the ability to capture scenarios that involve more complex coordination.

Building on the foundations of VDN, QMIX (Rashid et al., 2018) proposed a new architecture, a monotonic value function factorization, or a mixing network, that combined individual Q-values into a global Q-value. Additionally, there are hypernetworks involved in the algorithm that are functions of the global state, and the outputs of these layers enforce monotonicity. The monotonic constraint is defined as the following:

$$\frac{\partial Q_{tot}}{\partial Qa} \geq 0, \ \forall a$$

This constraint enforces that increasing an agent's Q-value will never decrease the team's overall Q-value. This constraint makes it possible for each agent to choose its own best action

independently, while still contributing to a high team score. However, the monotonic constraint also severely limits the different possibilities of coordination or tactics that agents can use that would otherwise further increase the joint Q-value. For example, if the optimal coordination calls for a tactical retreat or a sacrificial play (e.g., where one agent's Q-value would decrease but the overall joint Q-value would increase), QMIX fails to recognize this. The mixer strictly outputs positive weights (via absolute value operations) to guarantee agents only make actions that are guaranteed to be favorable for themselves which in turn raises the team Q-value.

Later works explored different modifications to the QMIX's (Rashid et. al, 2018) monotonicity, such as completely removing the constraint or developing new architectures for the mixing process. QTRAN (Son et al., 2019) explores the first option, as the monotonicity constraint was too restrictive. In place of the monotonicity constraint, QTRAN introduces a transformation function that, in theory, allows the joint action-value function to be expressed in additional ways that support any coordination pattern. Because the monotonic constraint is removed, to preserve the benefits of centralized training with decentralized execution, three loss constraints are introduced. These constraints are designed to ensure that agents choose the most favorable actions based on local Q-functions that will also lead to the same actions as those selected by the centralized joint Q-function. While QTRAN provides the most flexibility at a theoretical level, it suffers from the most training instability out of the QMIX variation and optimization challenges are also amplified due to all the interacting constraints.

Lastly, QPLEX (Wang et al., 2021) proposes an intermediate solution between the rigidity of QMIX's constraint and the QTRAN's training instability. The two main objectives behind QPLEX were to maintain the benefits of monotonicity but relax the constraint during advantage computation. To achieve this, QPLEX uses the idea of a dueling network or decomposition network that splits the global Q-value function into two output components: a baseline term and an advantage term. The base term is monotonic, while the advantage term captures non-monotonic interactions. An important aspect of this design is that the advantage term is regulated and will never overpower the base term, but will have a strong influence on it. The overall architectural design of QPLEX allows agents to learn more complex coordination patterns that would otherwise not be explored by QMIX. However, similar to QTRAN's weaknesses, introducing additional layers and parameters can be computationally costly and may not be scalable to other environments.

## 3 Methodology

It is clearly evident that while QMIX is powerful and scalable, the monotonicity constraint presents a real problem when applied to environments requiring complex coordination. The strict monotonicity works well in simple maps. However, for larger agent counts and heterogeneous unit compositions, the restrictiveness of the constraint limits coordination flexibility. Using principle ideas from QTRAN and QPLEX, particularly the addition of a correction term, RelaxedQMIX introduces a more flexible structure to the mixing process.

The RelaxedQMIX approach will explore a modification of the QMIX algorithm by incorporating a correction term that relaxes the monotonicity constraint.By taking a hybrid approach between QTRAN and QPLEX, RelaxedQMIX will extend the base QMIX output by adding a correction term. This new addition term will encourage agents to make decisions

beyond the traditional structure of QMIX to provide greater benefits for team coordination.

**Correction Network Design** The correction term can be implemented through a small augmentation to the original QMIX architecture. Specifically, the augmentation consists of two fully connected layers with ReLU activation functions. The inputs take in the concatenation of all agent Q-values and global state vectors. After the input is passed through the first layer, a state re-injection technique is applied. This technique concatenates the global state vectors again with the output from the first layer before it is passed into the second fully connected layer. After it is processed by the second layer, the resulting output provides the mixing network with more diverse information about the environment so that the model can make more informed decisions about when the corrections are beneficial. The final joint Q-value in RelaxedQMIX can be represented as:

$$Q_{tot} = Q_{mix}(Q_1, \ldots, Q_n) + \epsilon \times Q_{correction}(Q_1, \ldots, Q_n, s)$$

where $Q_{tot}$ is the final global Q-value, $Q_{mix}$ is the monotonic output of the original QMIX across all agents, and $\epsilon$ is a hyperparameter that scales the magnitude of the correction term. The hyperparameter allows the model to interpolate between strict monotonicity (when $\epsilon = 0$) and a more flexible and expressive formulation as $\epsilon$ increases.

**Stabilization Mechanism** To stabilize learning during the training process, the correction network is initialized with a very small random weight and the main purpose of this is to prevent the network from making too large of a correction early in the training. Then, an L2 regularization is applied which applies a penalty when the network makes a large correction. This effectively creates a budget constraint and together, these mechanics will encourage the model to primarily follow the base QMIX architecture and only break the monotonicity rule when there is a significant benefit to overall team coordination.

## 4 Implementation

For the purposes of reproducing this experiment, all training was conducted on the BU Shared Computing Cluster using NVIDIA V100 GPUs (or GPUs of equivalent processing power). It is assumed that Python Multi-Agent Reinforcement Learning (PyMARL), the StarCraft Multi-Agent Challenge (SMAC), and the Linux version of StarCraft II (for headless training) are properly installed and configured. PyMARL provides the libraries and framework and SMAC provides the environments for evaluating MARL algorithms, which range from simple to very complex scenarios.


*Figure 1*: Agents (red) must work together to beat opposing (blue) units. (*Left*) MMM2: 1 Medivac, 7 Marines, 2 Marauders vs 1 Medivac, 8 Marines, 3 Marauders (*Right*) corridor: 6 zealots vs 24 zerglings

**Experimental Setup** RelaxedQMIX was evaluated on four SMAC maps, consisting of two maps (*8m* and *2s3z*) categorized as easy and two maps (*MMM2* and *corridor*) as highly complex. The map designs are *8m* (8 Marines vs 8 Marines), a symmetric, homogeneous battle scenario; *2s3z* (2 Stalkers and 3 Zealots vs 2 Stalkers and 3 Zealots), a symmetric, heterogeneous scenario; *MMM2* (1 Medivac, 2 Marauders, and 7 Marines vs 1 Medivac, 3 Marauders, and 8 Marines), an asymmetric, heterogeneous scenario; and *corridor*, an asymmetric, heterogeneous, and tactical positioning scenario. The evaluation metrics that were used are win rate, training loss, and average reward. Successful metrics will maintain the high performance observed on simpler maps while improving learning outcomes on more challenging scenarios where QMIX typically fails to converge.

After fully implementing RelaxedQMIX into the PyMARL codebase, a few configurations were tested, and the most optimal settings were selected based on training stability and performance across the maps that were tested. For the base configuration, an epsilon-greedy exploration strategy was used, starting at 1.0 and annealing to 0.05 over 50,000 timesteps. A replay buffer of size 5000 was used, with target network updates scheduled every 200 episodes, and optimization was performed using the RMSprop optimizer with a learning rate of 0.0005. Each policy was trained for 2,050,000 timesteps, which approximates to 8–12 hours of training time depending on the map and system load.

**Hyperparameter Settings** RelaxedQMIX-specific hyperparameters were manually selected, as performing a full grid search would have led to prohibitively long training durations. A correction epsilon of 0.1 and a regularization coefficient of 0.01 were applied uniformly across all maps. To stabilize early-stage learning, the weights of the correction network were initialized with a small value (0.0001) to reduce the likelihood of large initial monotonicity violations.

## 5 Experimental Results

The observations of the training results for both QMIX and RelaxedQMIX are shown below, and there is strong evidence of learning on *MMM2* using the modified algorithm.

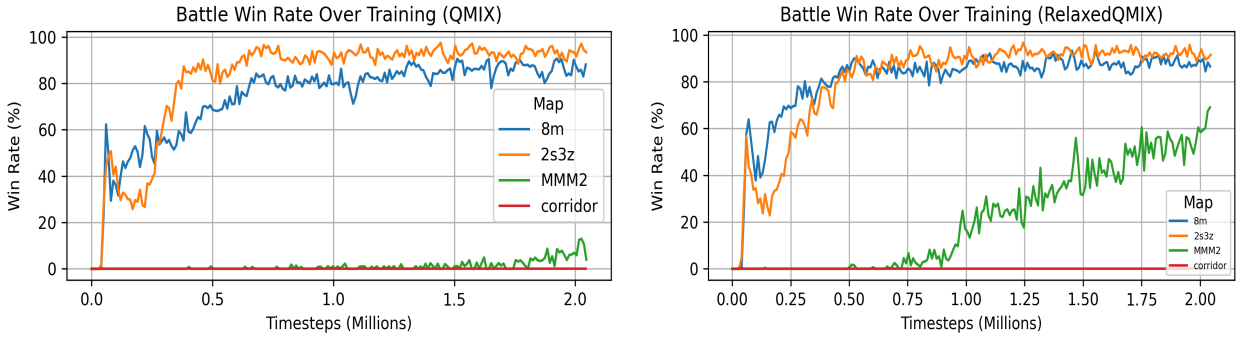


*Figure A (Left)*: Battle win rate of QMIX over 2,050,000 million time steps. *(Right)*: Battle win rate of RelaxedQMIX over 2,050,000 million time steps.

**Win Rate** Figure A shows the battle win rate over training. On the easy maps (*8m* and *2s3z*), both QMIX and RelaxedQMIX achieved high win rates above 80%. On the complex maps (*MMM2* and *corridor*), QMIX struggled to converge on both maps, while RelaxedQMIX achieved a

higher and continual improvement in win rate right to a peak of 70%. However, for both algorithms, *corridor* always failed to converge and maintain a 0% win rate.

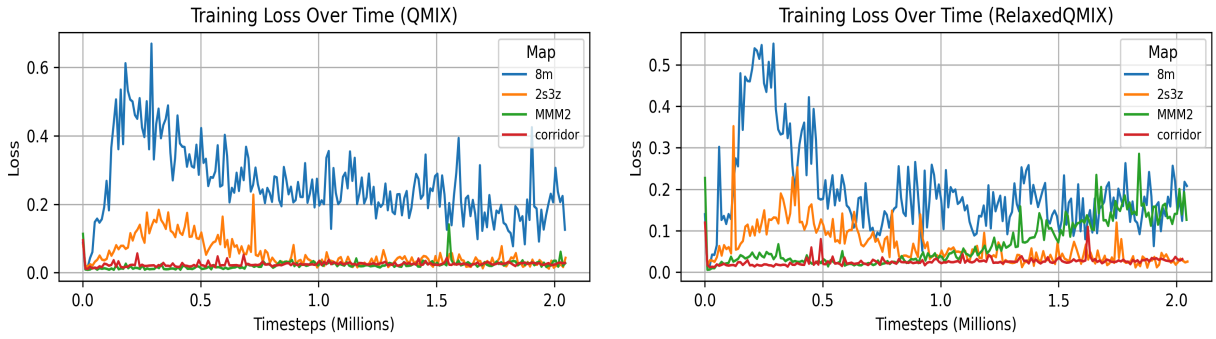


*Figure B (Left)*: Training loss of QMIX over 2,050,000 million time steps. *(Right)*: Training loss of RelaxedQMIX over 2,050,000 million time steps.

**Training Loss** Figure B presents training loss over time. On the easier maps, both QMIX and RelaxedQMIX display similar training loss curves. For *MMM2*, RelaxedQMIX instead exhibits a steady rise in training loss following 1 million time steps and peaks at approximately 0.2 at the end of the training process.

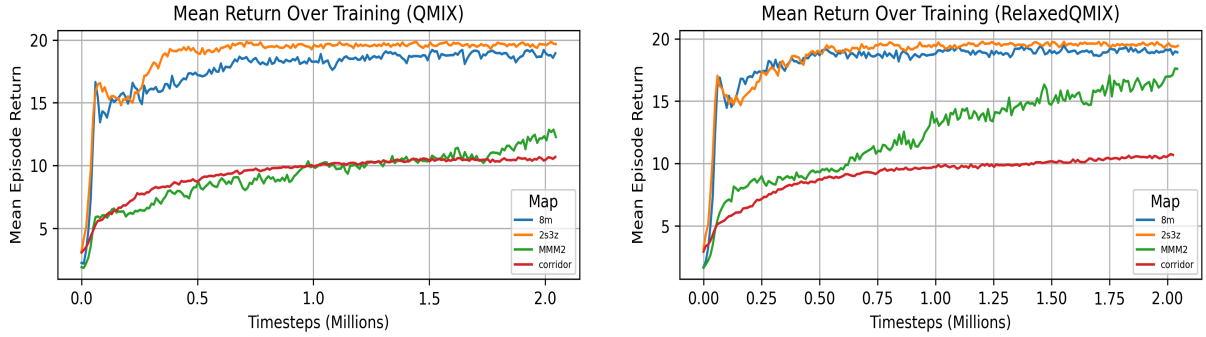


*Figure C (Left)*: Return mean of QMIX over 2,050,000 million time steps. *(Right)*: Return mean of RelaxedQMIX over 2,050,000 million time steps.

**Return Mean** Figure C shows the average episode reward throughout the training process. On most of the maps, the two algorithms have very similar reward trajectory; however, in the case of *MMM2*, RelaxedQMIX shows a noticeable increase in average reward after 1 million timesteps.

## 6 Analysis and Discussion

To evaluate and compare the performance between RelaxedQMIX and QMIX, four SMAC maps were selected: *8m, 2s3z*, *MMM2*, and *corridor*. These maps represent a wide range of coordination challenges, including homogeneous vs. heterogeneous agent types and symmetric vs. asymmetric team compositions. Each algorithm was trained for 2,050,000 environment timesteps, and performance was evaluated using three primary metrics: battle win rate, training loss, and mean episode return.

On the easier maps (8m and 2s3z), both QMIX and RelaxedQMIX converge to high win rates

and hover around 90%. This indicates that RelaxedQMIX is able to retain the benefits of QMIX and let the base function learn effective strategies in symmetric and/or homogeneous settings. However, on the more challenging maps (*MMM2* and *corridor*), performance begins to diverge, as QMIX is barely able to achieve a 10% win rate. RelaxedQMIX, however, starts to converge past the 1 million timestep mark, and an important observation to note is that there is consistent growth in win rate at the 2 million timestep mark, so there is a possibility for even higher performance given ample timesteps. This performance suggests that the added flexibility from the correction term helps agents better coordinate in heterogeneous, asymmetric scenarios. Lastly, *corridor* presents the highest degree of a heterogeneous and asymmetric setting, where either fine-tuning of parameters or a better mechanism must be put in place to capture positional awareness to maximize team coordination.

The training loss trends are similar for both QMIX and RelaxedQMIX on most maps, except for *MMM2*. In the case of *MMM2*, the training loss starts increasing after 1 million timesteps. This could suggest the correction term is beginning to influence the optimization process. Because the base component of QMIX could not converge, and once the correction term begins influencing the joint Q-value, the training loss increases as the algorithm continually tries to learn the best coordination strategy.

The average reward return serves as confirmation that the algorithm is learning new strategies once the monotonic constraint is relaxed. The increase in reward also correlates with the win rate gradually increasing. As with average return, both algorithms follow similar return trends on *8m*, *2s3z*, and *corridor*. On *MMM2,* however, RelaxedQMIX shows a sudden increase in return around the 1 million timestep mark, further supporting the hypothesis that the correction network is starting to take effect and enhance learning in environments where QMIX struggles. This trend is especially important given that average return captures partial progress toward a win, even in episodes where the task is not fully completed.

## 7 Conclusion

This project was successfully able to demonstrate that small modifications to QMIX's mixing architecture can yield greater results. The main concern of QMIX is the monotonic constraint, and it is sometimes too restrictive on maps that require more coordination. QMIX offers great performance in simple maps with homogeneous unit composition and symmetrical battles, and RelaxedQMIX was able to mirror the results. Most importantly, RelaxedQMIX achieved far greater performance on *MMM2*—a heterogeneous and asymmetrical battle—where QMIX struggled due to its restrictive nature. The main problem, however, is reflected in *corridor*, as RelaxedQMIX was not able to effectively learn the proper spatial coordination to hold a choke point at the narrow passage.

## 8 Future Work

There are a number of directions to further enhance the study conducted in this project. Due to the sensitivity of the algorithm to hyperparameter settings, careful hypertuning must be a priority. Instead of manually setting hyperparameters, a grid search could be conducted to fine-tune training and assess whether the algorithm is able to learn better coordination tactics on more difficult maps. Another step that could be taken is evaluating performance across all

SMAC maps with more timesteps. This would help identify the specific types of coordination challenges on which RelaxedQMIX struggles to converge on. While the relaxation of the monotonicity constraint is intended to support the learning of complex tactics such as sacrifice and tactical retreat, further testing is needed on maps that emphasize these behaviors to determine whether RelaxedQMIX can effectively capture such coordination patterns. Additionally, unlike a typical case study, where the timesteps would be around 10 million (Samvelyan et al. 2019), for the purposes of this project the timesteps were decreased to 2 million. Finally, as seen in the corridor map, the algorithm fails to explore tactical positioning as a key coordination strategy. Enhancing the architecture with graph-structured networks could be explored to determine whether agents can learn to use the environment more effectively to maximize team composition advantage.

**Special Note**

All reproducible code and links to videos can be found in this link:
`https://github.com/miruyoun/PyMARL_RL_Project`

**References**


Kraemer, L. and Banerjee, B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing, 190:82–94, 2016.

Oliehoek, F. A., Spaan, M. T. J., and Vlassis, N. Optimal and Approximate Q-value Functions for Decentralized POMDPs. Journal of Artificial Intelligence Research, 32 (1):289–353, 2008.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2018). *QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning*. In *Proceedings of the 35th International Conference on Machine Learning (ICML)* (Vol. 80, pp. 4295–4304). PMLR. https://arxiv.org/abs/1803.11485

Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., & Whiteson, S. (2019). *The StarCraft Multi-Agent Challenge (SMAC): Decentralized micromanagement*. arXiv preprint arXiv:1902.04043.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., & Yi, Y. (2019). QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *In Proceedings of the 36th International Conference on Machine Learning* (ICML 2019), PMLR 97:5887–5896.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., ... & Graepel, T. (2017). Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. *In Proceedings of the Tenth International Conference on Machine Learning* (pp. 330–337). Morgan Kaufmann.

Wang, J., Ren, Z., Liu, T., Yu, Y., & Zhang, C. (2020). QPLEX: Duplex dueling multi-agent Q-learning. *arXiv preprint arXiv:2008.01062*.