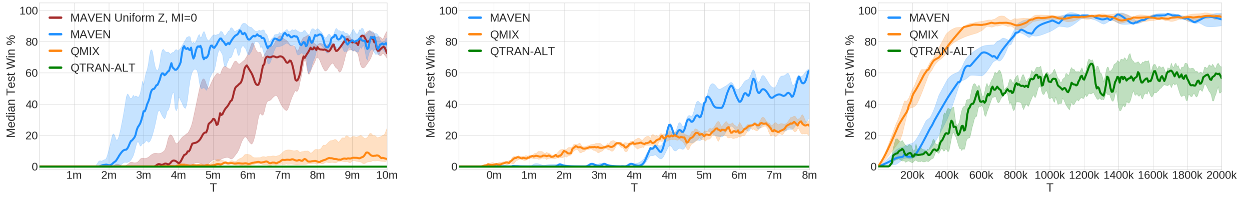
We thank all the reviewers for their feedback. All reviewers are concerned whether we substantially outperform QMIX. Since StarCraft II experiments take a long time, we could not include all the results in the submission. However, we have now obtained results for most of the SMAC maps [The StarCraft Multi-Agent Challenge, Samvelyan et al 2019], which Samvelyan et al. have classified as Easy, Hard & Super Hard. We have also added a recent method QTRAN [Son et al, ICML 2019] as another baseline. Results on several maps are shown below. Note in particular two of the **Super Hard** maps in SMAC: Corridor (Fig. 1(a)) and 6h_vs_8z (Fig. 1(b)). The plots show that MAVEN performs substantially better than all alternate approaches. We observe this trend on rest of **Super Hard** maps in SMAC with performance similar to QMIX on **Hard** and **Easy** maps. **Thus MAVEN performs better as difficulty increases**. This, along with the adaptation experiment in Fig 4(b) of the paper, strongly support the importance of the committed exploration offered by MAVEN for decentralised MARL. We will include all the new results in the final version. Furthermore, QTRAN does not yield satisfactory performance on most SMAC maps (0% win rate). The map on which it performs best is 2s3z Fig. 1(c), an **Easy** map, where it is still worse than QMIX and MAVEN. We believe this is because QTRAN tries to enforce optimal decentralisation using relaxed L2 penalties which are not sufficient for challenging domains.

In terms of theoretical contributions, **we are the first to analyse the effects of representational constraints on exploration, a problem which is unique to decentralised MARL**. To this end, we also quantify the suboptimality of QMIX, the current SOTA under *uniform* and ϵ-*greedy* exploration. We next address the important individual reviewer comments. We refer to the specific comment by **C** followed by its number in their review:



(a) `corridor` (b) `6h_vs_8z` (c) `2s3z`

Reviewer 1 – C1: Fully decomposed methods can in principle represent the optimal policy but poor exploration can cause them to converge to suboptimal policies due to representational constraints like monotonicity. MAVEN solves this by exploring different joint behaviour modes. **C3:** We provide that ablation for 3s5z map in App C.2 Fig 7(b) and for the more illustrative corridor map here in Fig. 1(a) **red line MAVEN Uniform Z, MI=0**. **C4:** We did not encounter any problem with stability as the parameters corresponding to the objectives are disjoint. **C5:** Yes, ϵ-greedy is used to regularise the training of neural nets involved in Q-value estimation.

Reviewer 2 – C1: We disagree with your evaluation about the uniqueness of the problem. **Inefficient exploration hurts decentralised MARL agents, not just in the usual way of single agent RL, but more importantly it interacts with the representational constraints necessary for decentralisation and can push the algorithm towards strictly suboptimal policies**. While single agent RL can avoid convergence to suboptimal policies using various strategies like increasing the exploration rate (ϵ), ensuring optimality in the limit, this is not the case with decentralised MARL. As **Theorem 2 shows, increased exploration can in fact reduce the chance of finding optimal behaviour**; this result is in **stark contrast to single agent RL**. Existing methods for single-agent exploration do not address this problem. **C2:** Not only is MAVEN different from DIAYN in the use case, it also enforces **action diversification** which means the agents jointly learn to solve the task is many different ways; **this is how MAVEN prevents suboptimality from representational constraints**, DIAYN is concerned only with discovering new states. Furthermore, DIAYN trains on diversity rewards using RL whereas we train on them via gradient ascent. We will clarify these points in the final version and add more discussion about DIAYN.

Reviewer 3 – Note that QTRAN was published 9 days before the submission deadline, so it was infeasible to empirically test it. Now that we have tested QTRAN on SMAC, it clearly performs poorly compared to QMIX & MAVEN on challenging domains like StarCraft II. In fact, there is **no empirical evidence that QTRAN can perform well beyond toy domains**. The QTRAN paper evaluates it only on toy domains and uses 10 million training steps, the same number we use for our much more complex domains. While Theorem 1 in the QTRAN paper guarantees optimal decentralisation, it imposes $\mathcal{O}(|S||A|^n)$ **constraints on the optimisation problem** involved, where $|S|, |A|$ are the sizes of state and action spaces and n is the number of agents. This is **computationally intractable** to solve in discrete state-action spaces and is impossible given continuous state-action spaces. The authors of QTRAN propose two algorithms (QTRAN-base & alt) which relax these constraints using two L2 penalties. Thus **QTRAN does not overcome QMIX's limitations as it deviates from the exact solution**. We will discuss these points in the final version. **C1:** As previously mentioned, there is no evidence that QTRAN is useful for challenging MARL domains but we think the L2 penalties involved can still be included in MAVEN as auxiliary loss. **C2:** The hierarchical policy is on the latent space variables which in turn control joint behaviour of agents for the entire episode. **C3:** The m-step matrix game is a 2-player multi-step game; the actions for the players are the row and column indices of the matrix to pick at each step; the goal is to maximise total reward; we have marked the initial state in Fig 2(a). We will make sure to improve the clarity of the algorithm section (4 Methodology).