We thank all the reviewers for their constructive feedback on improving the paper. We first summarize the new experiments done during the rebuttal period. Due to the limited time/compute (and the need to run multiple seeds), we were able to complete all runs for two MuJoCo environments (*Hopper*, *Half-Cheetah*). We run for 1M timesteps and report the final policy performance (mean ± std) over the seeds. Due to space constraints, we omit the learning curves but would add them to the revision, along with data for all the other tasks. To put the numbers below in context, SAC (IRCR) achieves 8275±1279 (for Half-Cheetah) and 3214±279 (for Hopper) in the episodic reward setting (Figure 3).

	EXP. 1		EXP. 2	EXP. 3	EXP. 4	
	SAC (env. rewards) n-step returns (n=10)	SAC (env. rewards) λ-returns (λ=0.9)	Distributional-SAC (env. rewards)	SAC (env. rewards) State-augmentation	TD3 (env. rewards)	TD3 (IRCR)
Hopper	626±281	566±101	292±31	358±123	235±37	2981±114
Half-Cheetah	−42±29	−112±206	−24±16	−82±36	−123±50	6844±918

	EXP. 5		
	SAC (env. rewards)	SAC (IRCR)	Random Policy
Robotic Arm	104±4	160±7	90±11

[R1.] *Q. Are exploration and credit assignment (due to delayed rewards) the same?* Exploration and credit assignment are two distinct fundamental problems in RL. The former deals with the *discovery* of new useful information, the latter is about efficiently incorporating this information for learning a robust policy. In hard exploration problems, an agent typically obtains zero rewards in each episode unless an exploration impetus is given, whereas in our setting, a reward signal is readily provided to the agent at the end of *every* episode. The focus, therefore, is to train effectively from this delayed feedback and improve upon the credit assignment. We agree that it's important to clarify this distinction and would expand on this further in the revision. *Q. n-step returns and mixed Monte-Carlo?* We include these baselines now; please see EXP. 1 for the results. SAC (λ-returns) augments the Q-function training with TD(λ), which interpolates nicely between 1-step TD and MC returns based on the value of λ. We tested with n={3,10,20} and λ={0.5,0.9,0.95,1.0}. Our conjecture for their low scores with episodic rewards is that the reward delay exacerbates the variance is MC, and bias in TD, to such an extent that using them separately, or mixing them via interpolation, is not sufficient to alleviate the problems in value estimation for credit assignment. IRCR takes a different approach of learning guidance rewards for each time-step, and integrates well with 1-step TD (due to the bias reduction afforded by the guidance rewards).

[R2.] *Q. Limitations of IRCR?* Since the reward that the agent optimizes for is different from the original task reward, and is coupled to a behavioral policy β, for a given MDP, it might be possible to design an adversarial β such that optimizing for the resultant guidance rewards leads to unintended behaviors (as per the task rewards). Although not visible in our empirical evaluation, a limitation of IRCR is that a careful adaptation of β could be crucial in some domains to avoid this. For instance, if β gets stuck in some region of the state-action-space, the learning agent may also get trapped in a local optimum due to *deceptive* guidance rewards. Combining IRCR with methods that explicitly incentivize exploration is a promising approach. We'll include this in the revision. *Q. Unintended output in provided example?* For the given scenario, using IRCR should not make the agent learn $a \rightarrow b \rightarrow e$. Since $a \rightarrow b \rightarrow c$ and $a \rightarrow d \rightarrow e$ are the high return trajectories, the guidance reward for the $b \rightarrow c$ transition should be higher relative to $b \rightarrow e$, making it preferable. *Q. Selection of Baselines?* Some of the methods in Related-works (EBU, HCA, NEC) are proposed for tasks with a *discrete* action-space as they utilize networks that scale with the number of actions. Our Reward-Regression baseline tackles credit assignment in a manner quite similar to RUDDER (as described in 4.1); the available RUDDER code does not handle continuous actions. Self-Imitation is a recent concept introduced to better handle tasks with delayed rewards. *Q. IRCR if there are indeed dense rewards?* In this case, we observe that IRCR reaches similar asymptotic performance compared to the baseline learned with dense rewards but its sample-efficiency is worse; this is because the guidance rewards become more representative gradually, as more data is collected in the MDP.

[R3.] *Q. Separate partial observability from reward delay?* We have added a SAC baseline where the state-space is augmented, $\tilde{s}_t = \{s_t, t, \text{done}, \sum_{\leq t} \gamma^k r_k\}$. This restores the Markovian property of the reward function. The low score of this baseline (EXP. 3) points to delayed rewards being the central issue. *Q. C51 baseline for MuJoCo; Reward-regression (RR) on all tasks?* We have added a distributional variant of SAC (EXP. 2). For RR, unfortunately, the code is not open-source, and the authors were unwilling to run additional experiments for us. *Q. Discuss failure modes/limitations?* Please see response to **[R2.]** *Q. Comparison with Eligibility traces?* Thanks for the references and the nice interpretation *w.r.t.* a replacing trace. We'll discuss this in the revision. The baseline in EXP. 1 (λ-returns) is also relevant here due to the equivalence between TD(λ) (forward-view) and Eligibility traces (backward-view).

[R4.] *Q. More domains?* We have added a robotic arm task with episodic rewards: peg-insertion with a 7 DoF Sawyer robot (EXP. 5). We were unable to include further tasks due to limited time/compute and the other rebuttal experiments. *Q. TD3 and TD3 (IRCR)?* We have added these (EXP. 4). *Q. Why multi-agent?* Credit assignment is challenging in the Rover domain because if the agents are outside the observation radius of all POIs – which is typically many timesteps in an episode – then no reward is achieved. Just like single-agent RL, this hampers learning when TD-based MARL is employed. The results on this task show that our approach is generally applicable (*i.e.* not specific to single-agent tasks). No changes were required for the mechanism of calculating the guidance rewards; the architectural modifications are included in Appendix A.2. *Q. Why IRCR works well with data from outdated policies in replay buffer?* Our intuition here is that the return normalization (Line 18, Algo. 2) helps to stabilize training. Specifically, if the trajectory returns from the outdated policies are low compared to the best return seen thus far (this happens gradually over time), the magnitude of guidance rewards for those state-action pairs is low. Hence the agent refrains from visiting those regions.