Thank you for the constructive feedback. We are encouraged that the reviewers find our approach to be an interesting way to encode the underlying graph [**R2**] and a scalable approach to solving more complex domains [**R3**, **R4**] that results in considerable improvements [**R1**, **R2**, **R3**, **R4**] and compares well with existing methods [**R2**, **R3**, **R4**]. We will address minor writing suggestions and incorporate the additional references. We now address some specific questions and present a couple more results which will be included in the paper.

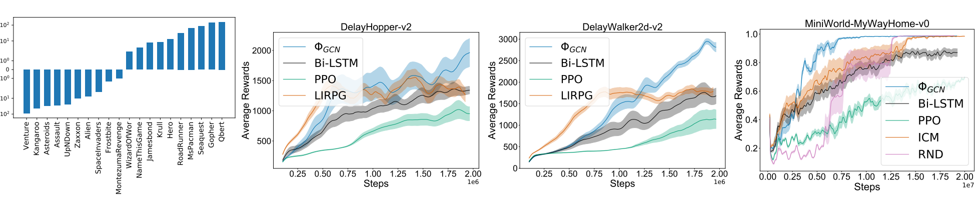
[**R1**,**R3**] **Time/computational complexity results.** We performed *additional analysis* in **Table 1** where we evaluated each baseline on the Atari domain (other domains follow similar trend). We did these evaluations on a single V100 GPU, 8 CPUs and 40GB of RAM. The time taken (in frames-per-second (FPS), so high is good) for our approach Φ_{GCN} is very similar to the PPO baseline, only slightly slower. We also compare favourably with respect to the RND, ICM, LIRPG and Bi-LSTM [**R4**] baselines. We believe this good performance stems directly from our sampling strategy that is minimal yet effective.

[**R2**,**R4**] **Continuous control.** Although continuous control presents challenges, our current algorithm, which relies on sampling trajectories rather than constructing the full graph, is still an effective approach **as shown in additional results for continuous environments provided below**. We conducted these experiments on the delayed Mujoco domain where the extrinsic reward is rendered sparse by accumulating it over 20 steps before it is being provided to the agent. We averaged the results over 10 random seeds. **Figure 1b-c** shows that our approach still provides significant improvements over the PPO, LIRPG and Bi-LSTM baselines. In general, using graph-based learning in continuous domains can be tackled in various ways, such as using grid cell-like constructs (which we discuss briefly in Sec.3.1), or combine our sampling strategy with a model-based approach, in which we would roll out the model from states observed on a trajectory. We will add more discussion on this to the future work section.

[**R4**] **On the advantage of GCN vs RNN.** In order to answer this question, **we performed additional experiments** on the MiniWorld and Mujoco domains to verify whether a Bi-LSTM, together with the GCN's loss function, would perform similarly. We chose a Bi-LSTM because it can propagate information both forward and backward in time, which is better suited to our problem. In **Figure 1d** we see that although there is improvement over the PPO baseline, the Bi-LSTM does not perform as well as the GCN based reward shaping. Moreover, in **Table 1** we notice that the Bi-LSTM runs considerably slower than the PPO and GCN baseline. We believe that GCNs provide an advantage (even for sampled trajectories) due to their architectural/structural bias, which has an important property: **local connectivity**. In contrast, an RNN's output would depend on potentially all past states (in the case of LSTM/GRU this depends on the weights themselves), and the bias is towards temporal connectivity on a particular trajectory, not local connectivity. Because we essentially want to make predictions on the state space graph, local connectivity leads to better results. We think a secondary factor is the fact that GCNs avoid exploding/vanishing gradients.

[**R1**] : **Inference or learning:** our paper focuses on both. Although $P(O|S)$ is clearly defined, we do not have access to it since we do not have access to the MDP's reward function. We hope to clear this misunderstanding by moving the algorithm box from appendix A.2 to the main paper. [**R1**] suggests that "it should not be as easy as stated in the paper" but does not expand on this reasoning. We would like to argue that our sampling strategy is effective, scalable and inexpensive as verified through various empirical evaluations (in the paper and in this rebuttal).

[**R3**,**R4**] **Related work and additional experiments:** We will gladly incorporate the suggested related works. Since **LIRPG** is indeed a valuable baseline and has online code, **we performed additional experiments** on the same set of 20 games from the Atari domain. In **Figure 1a** we plot the relative improvement with respect to PPO and see that LIRPG achieves overall good but mixed results. In some environments it achieves good improvements, whereas in a handful others the score is almost reduced to zero (note that our approach did not dramatically degrade performance). An important issue related to LIRPG is its wall-clock time performance (in **Table 1**) which is a considerable roadblock in terms of scalability and practicality.



(a) LIRPG on Atari (b) DelayHopper-v2 (c) DelayWalker2d-v2 (d) RNN reward shaping

Method	FPS
PPO	1126
Φ_{GCN}	1054
Bi-LSTM	815
RND	987
ICM	912
LIRPG	280

Table 1: Frames-Per-Second (FPS) on Atari

Figure 1: Additional experiments