Combining Graph-Based Planning and Deep Reinforcement Learning

Masters Thesis

Travis Hammond

September 2024

Outline

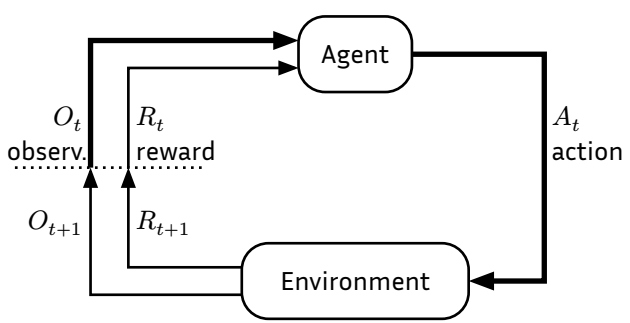
Introduction

The Building Blocks: Deep Reinforcement Learning

Learn a *global solution* over all states

- i.e. a value or action-value function

Operates on atomic (low-level) actions



Fails at long-horizon planning

- i.e. many actions into the future

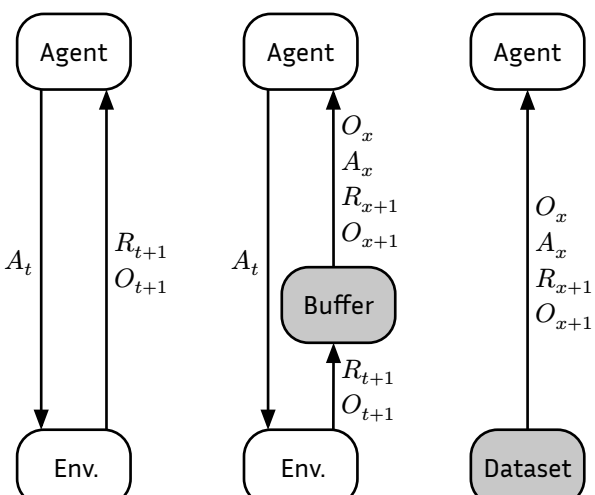
Can we fix that?

The Building Blocks: Deep Reinforcement Learning

Incoming data stream to train the deep neural network violates the **i.i.d** assumption:

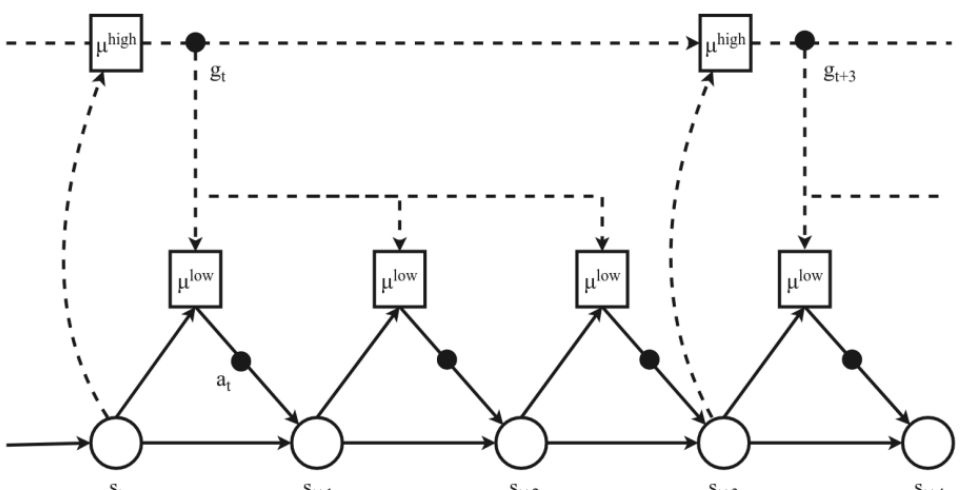
- Distribution determined by current policy

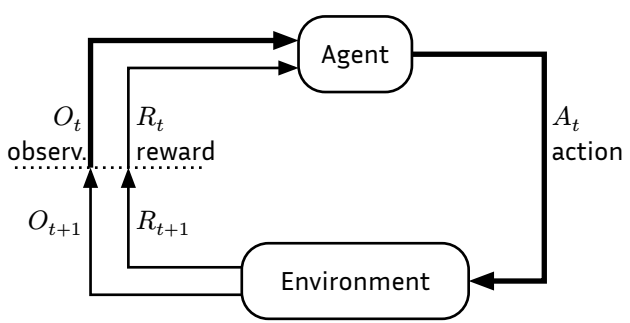
Replay Buffers are commonly used to partially restore this assumption and to improve sample efficiency.



The Building Blocks: Hierarchical RL

Abstract over time, i.e. sequences of actions





[1]

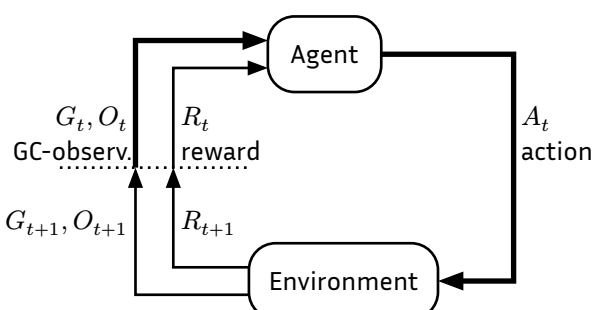
The Building Blocks: Hierarchical RL

An elegant solution: parameterize the goal!

- Goal-Conditioned RL (aka UVFA)

Some challenges:

- Even more general -> harder to train
  - Even shorter horizon!
- How to generate subgoals?
  - Suitable level of abstraction

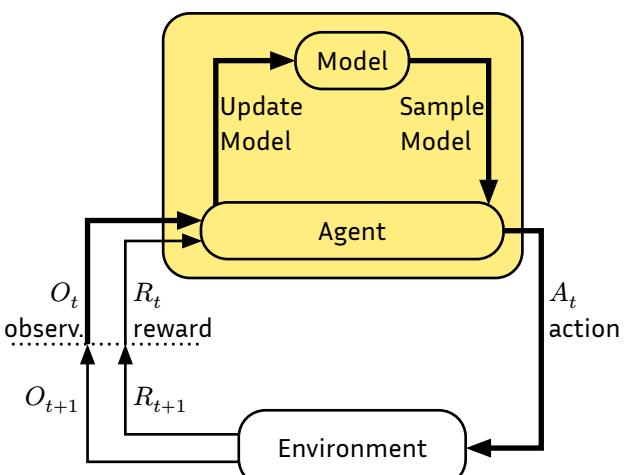


The Building Blocks: Model-Based RL

Idea is to learn a model of the environment

With *reversible access*

- Can simulate taking actions
- Can repeatedly plan forward from any state



Benefits

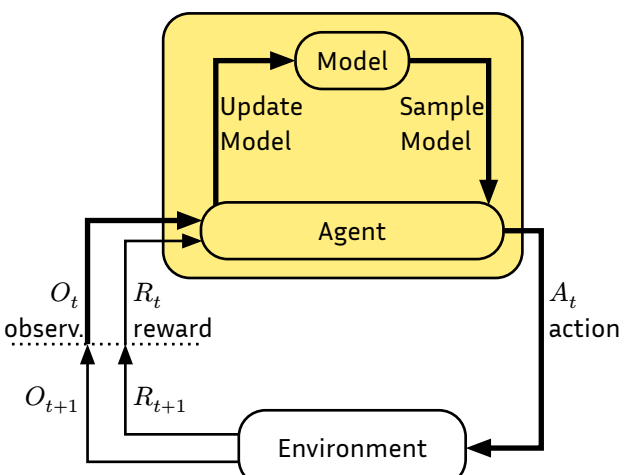
- Better sample efficiency
- Better at long-horizon tasks
- Interpretable Model

The Building Blocks: Model-Based RL

Idea is to learn a model of the environment

With *reversible access*

- Can simulate taking actions
- Can repeatedly plan forward from any state



Benefits only IF the model is good, otherwise:

- Model sampling becomes the burden
- Worse performance

Bad model –> Bad influence

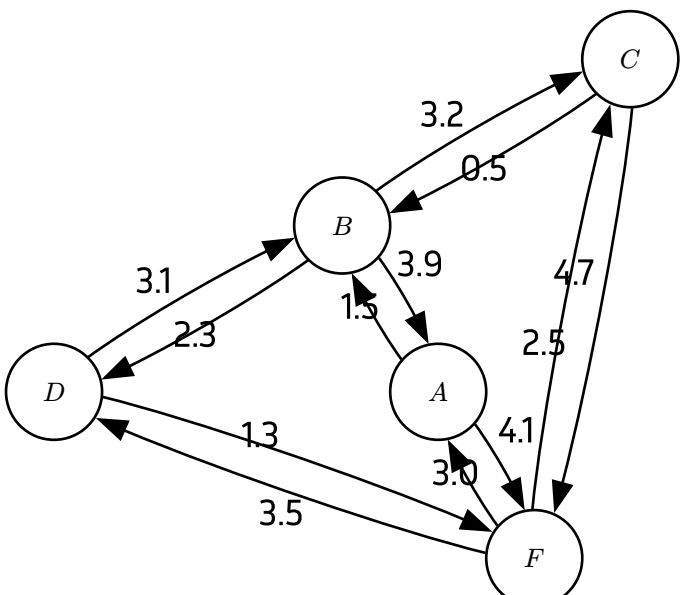
The Building Blocks: Graph-Based Planning

Meanwhile, we all remember Dijkstra?

- Long-horizon planning –> No problem!

Nice properties:

- Stable performance: $O(|E| + |V| \log |V|)$
- Guarantees (completeness & optimality)



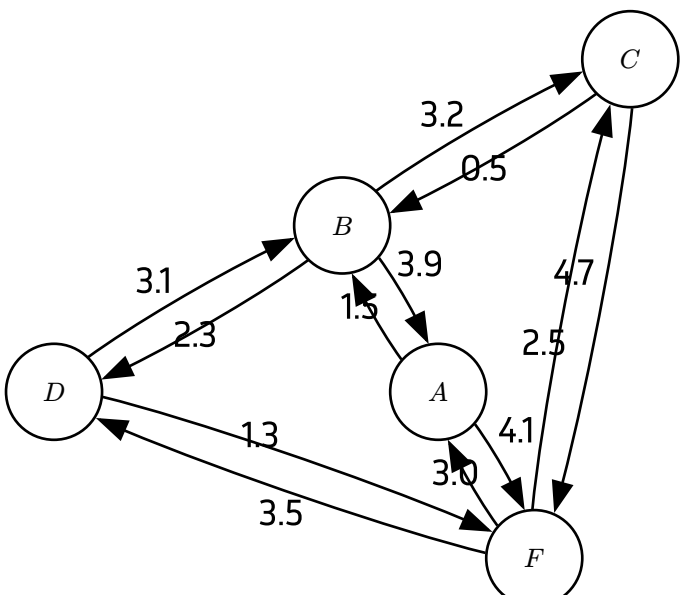
The Building Blocks: Graph-Based Planning

Meanwhile, we all remember Dijkstra?

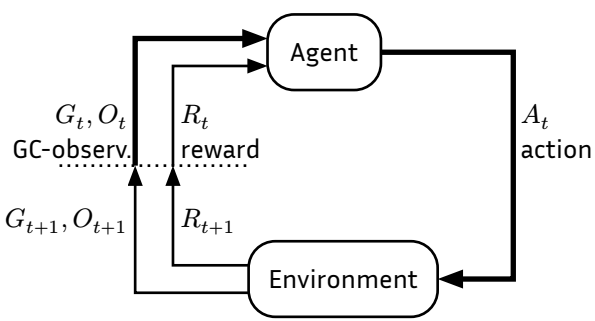
- Long-horizon planning –> No problem!

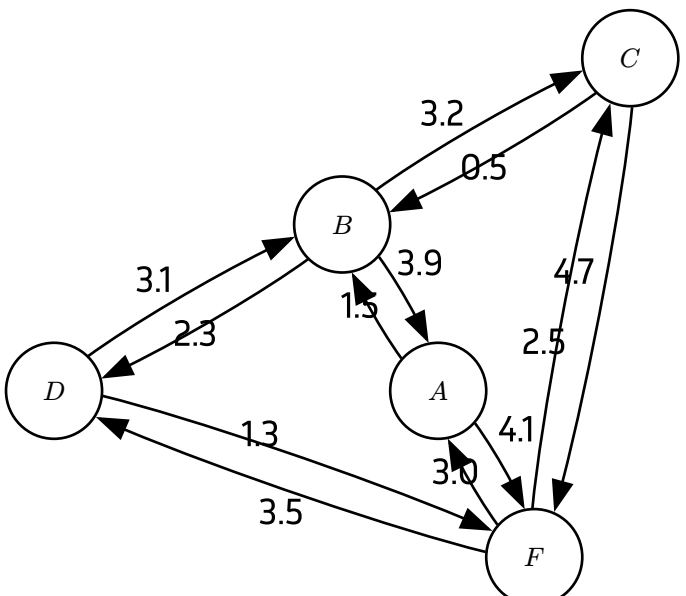
But...

- Requires handcrafted graph-representation
  - Set of nodes
  - Set of weights / edges
- How to actually move between nodes?



The Idea: Combine Graph-Based Planning and DRL





Low-level Controller: Reaches short-horizon subgoals via goal-conditioned RL.

High-level Controller: Solves long-horizon tasks via sequencess of short-horizon subgoals.

Methods

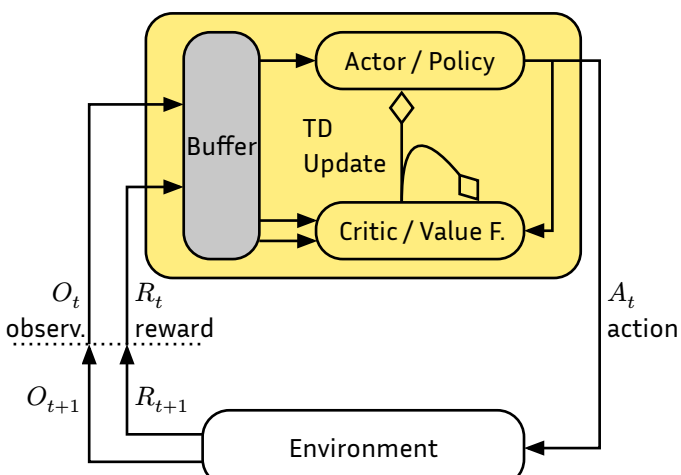
Actor-Critic Architecture (DDPG)

Components:

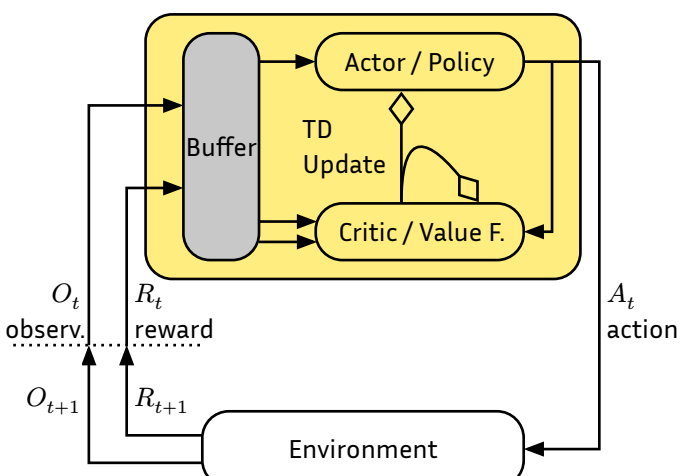
- Replay Buffer
- Actor(State) –> Action
- Critic(State, Action) –> Reward

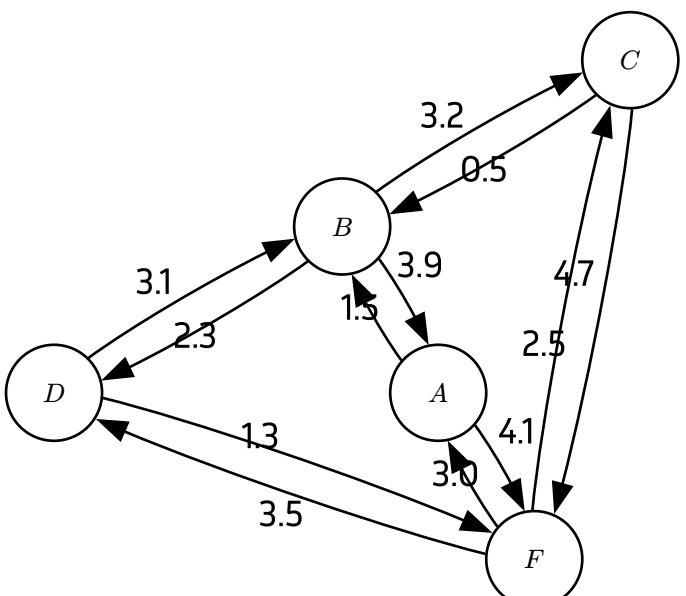
Benefits:

- Loosely coupled –> allows multiple actors
- Can handle continuous environments
- Training stability

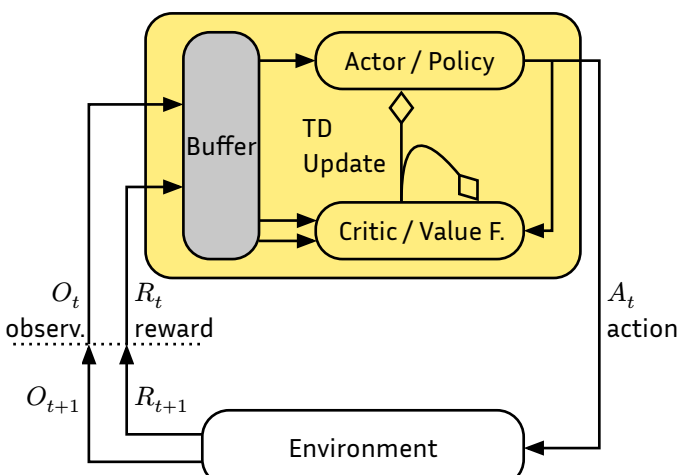


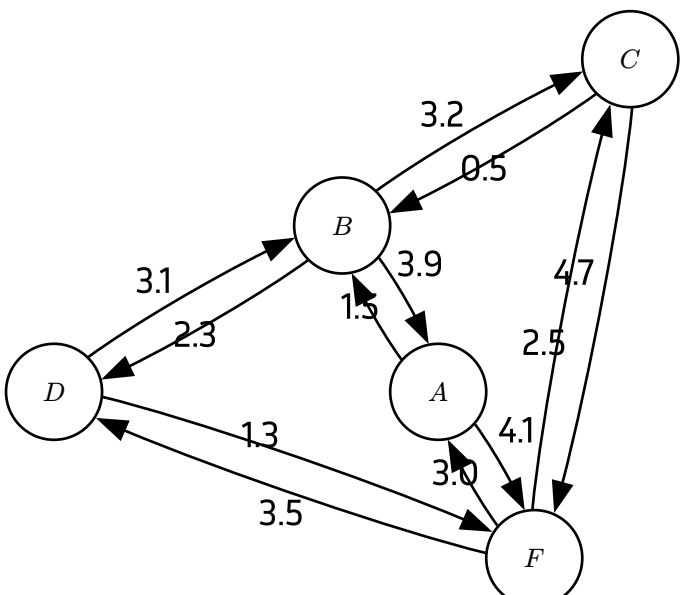
Big Idea: The Critic is the Distance Measure



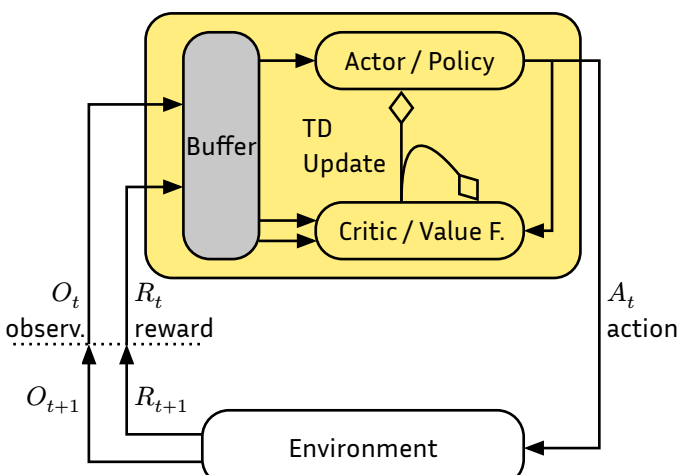


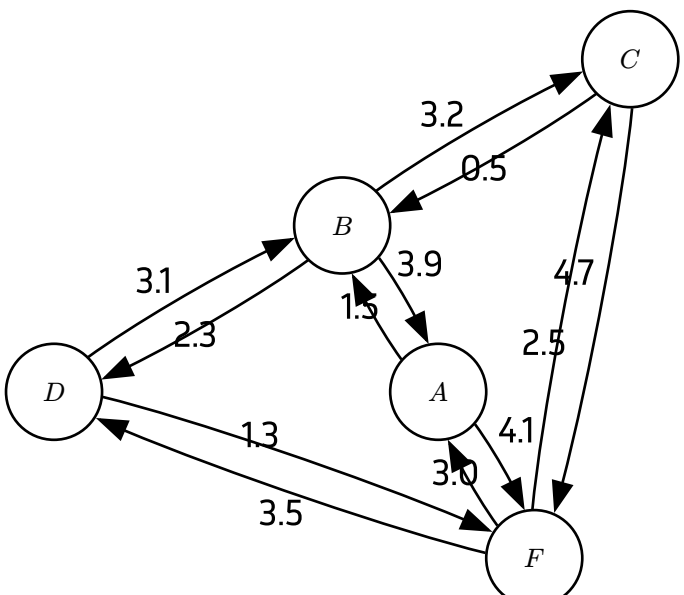
Big Idea: The Actor is the Controller



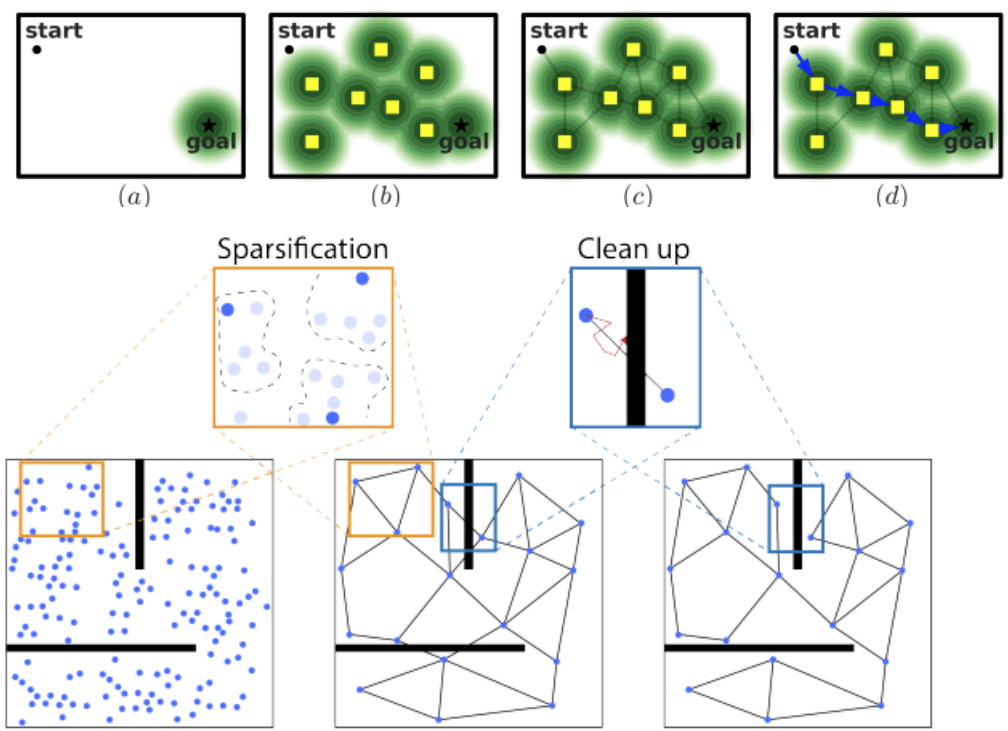


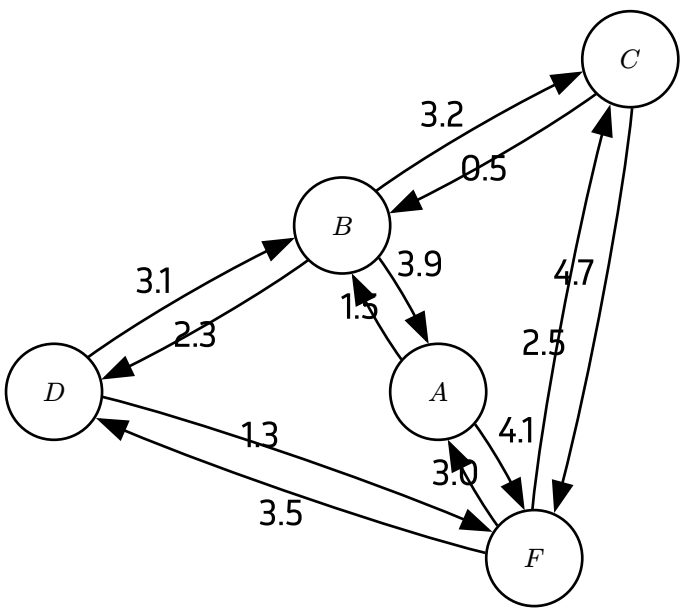
Big Idea: The Buffer contains the Nodes





Challenge: Graph Sparsification





[2], [3]

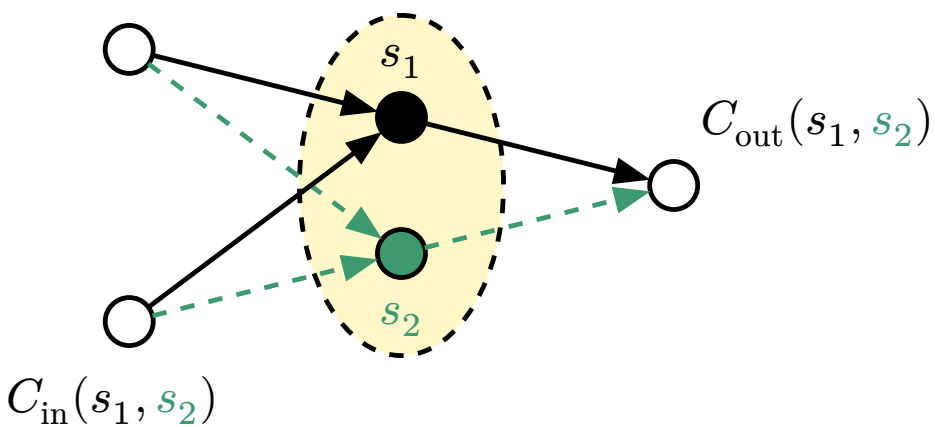
Challenge: Graph Sparsification

- Online sparsification algorithm $O(|V|^2)$ [3]
- For any asymmetric distance function
- Proof of error bound in shortest-paths

Two states are redundant if they are interchangeable as both starting states and goal states.

$$C_{\text{out}}(s_1, s_2) = \max_{\omega} |d(s_1, \omega) - d(s_2, \omega)| \leq \tau$$

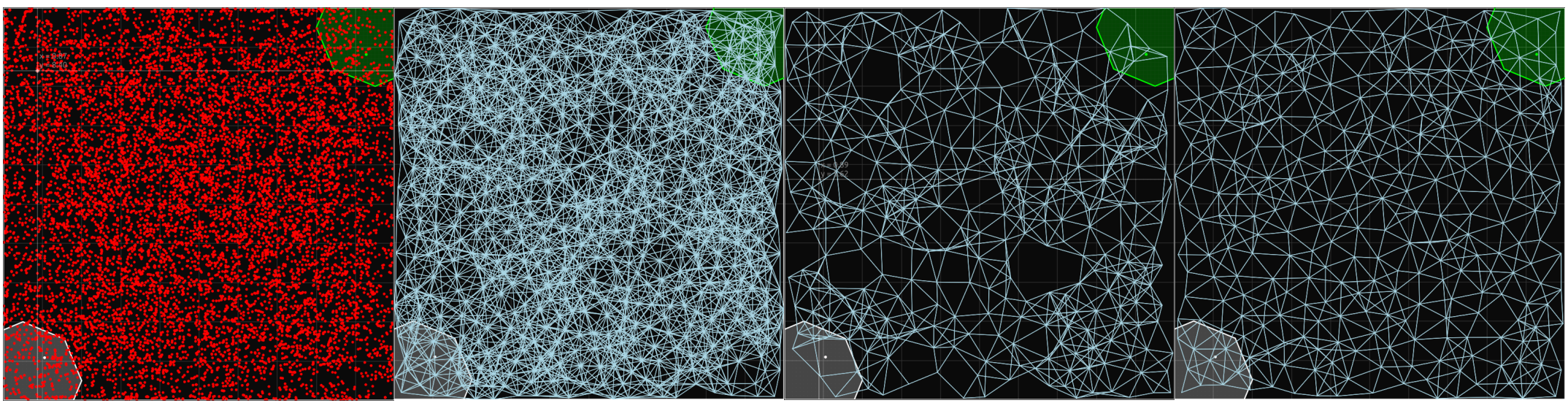
$$C_{\text{in}}(s_1, s_2) = \max_{\omega} |d(\omega, s_1) - d(\omega, s_2)| \leq \tau$$



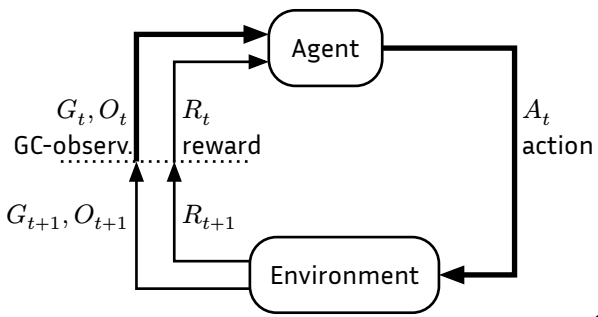
[3]

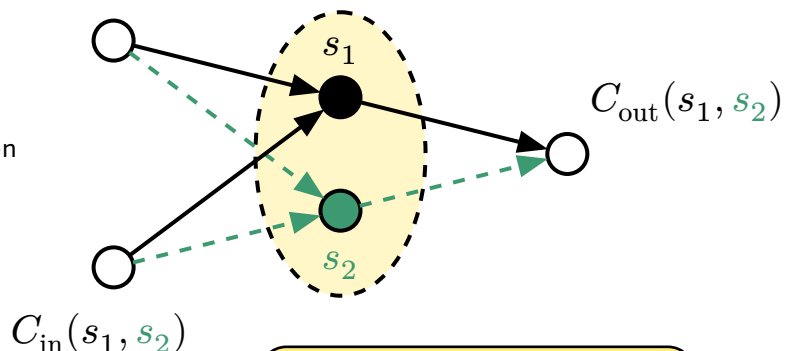
Challenge: Graph Sparsification

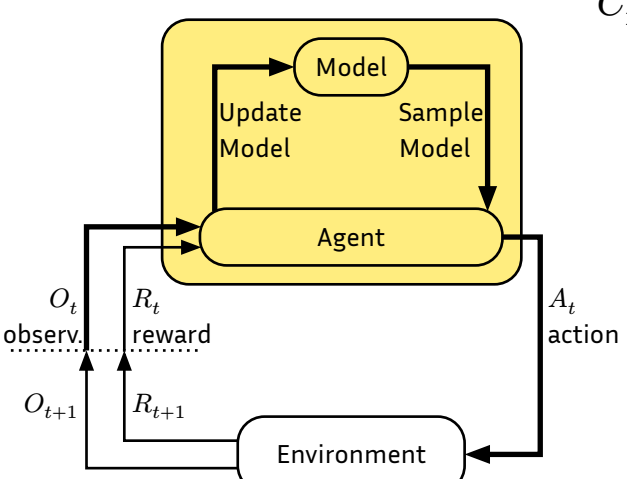
A full Replay Buffer sparsified with $\tau \in \{0.32, 0.40, 0.48\}$

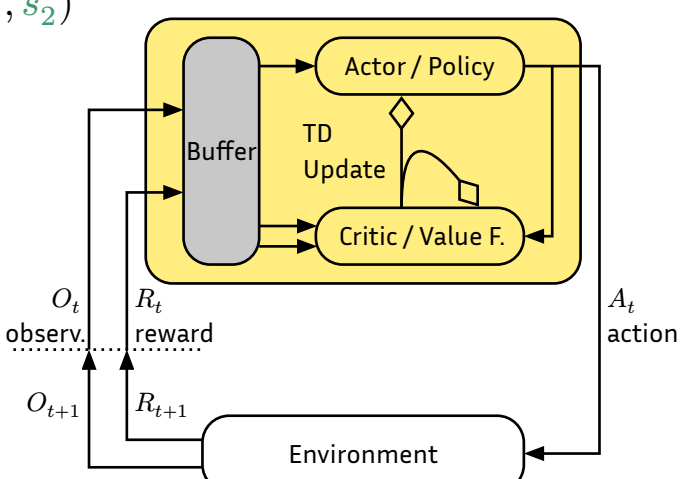


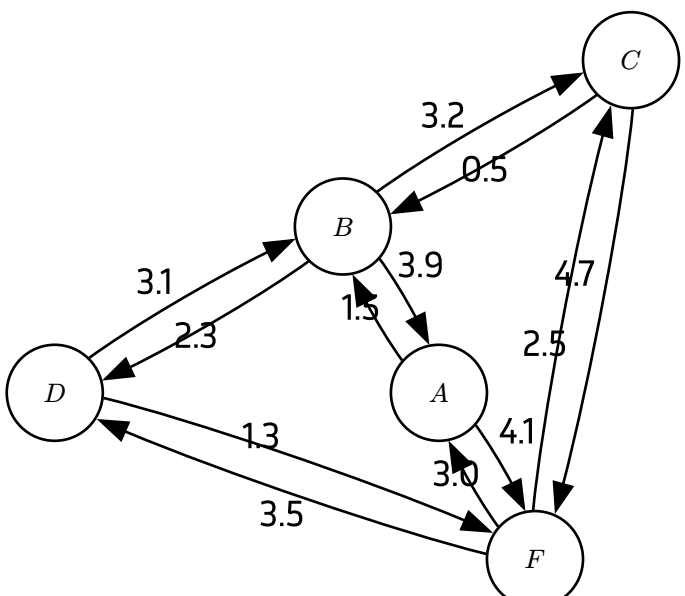
Building Block Overview











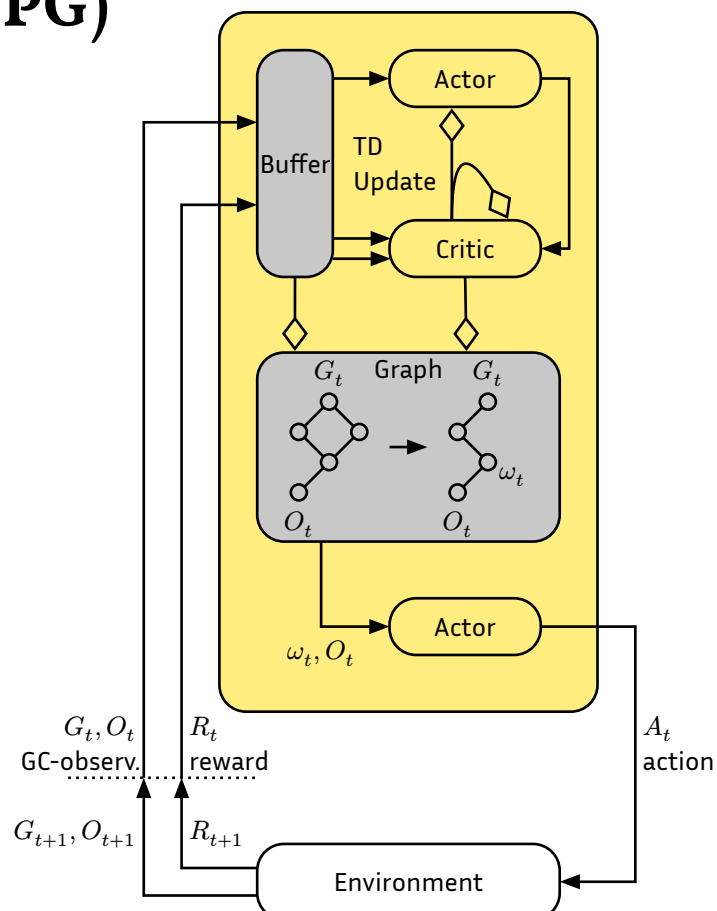
HGB-DDPG (Hierachical-Graph-Based DDPG)

Graph representation:

- Nodes from the Buffer
- Edge-weights from the Critic
  - actually we provide the true distances :(
- Sparsified via SGM + MAXDIST

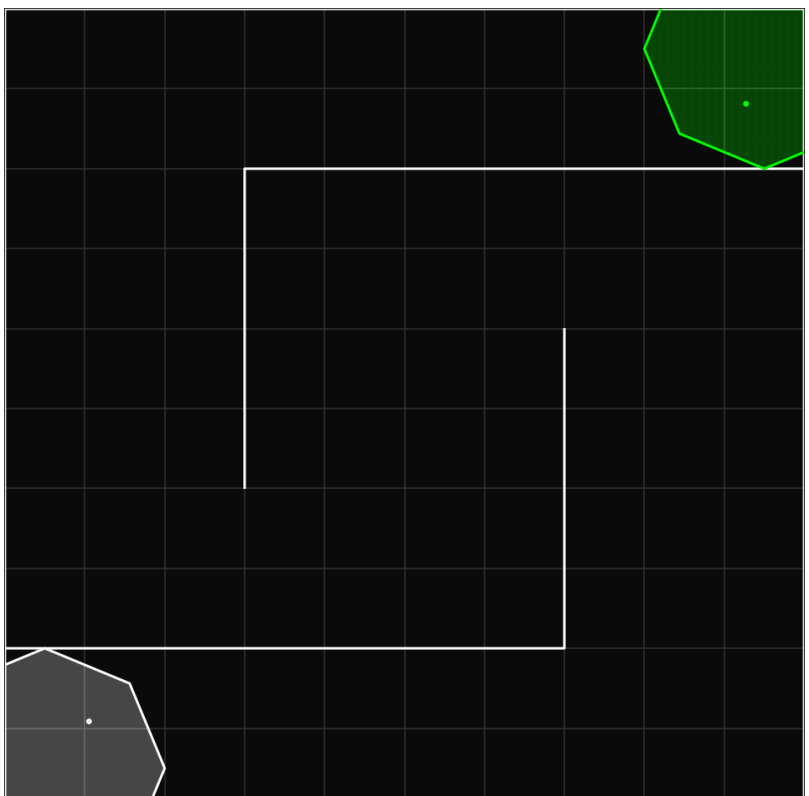
Cleanup:

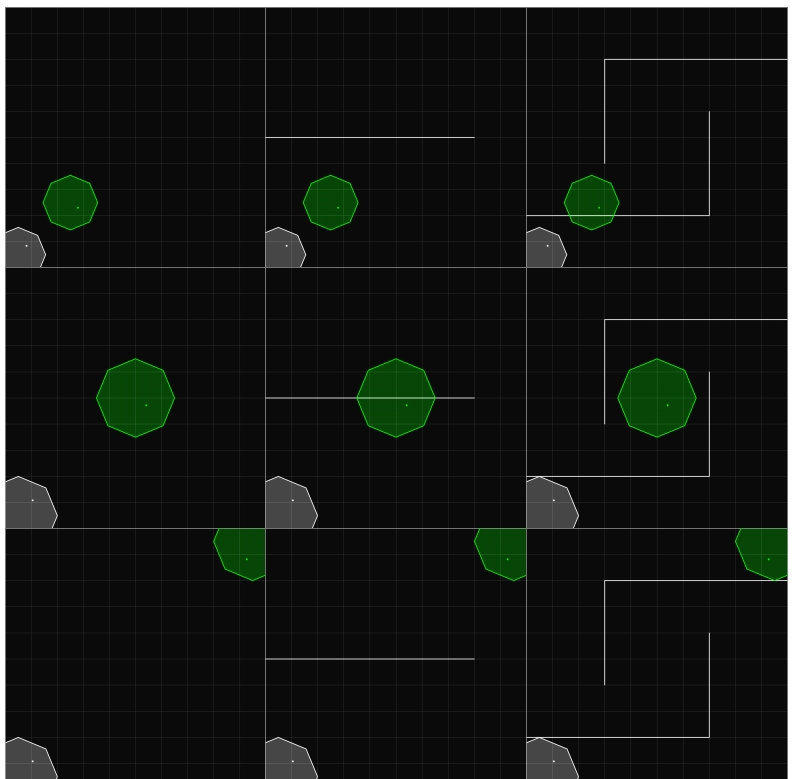
- Edges that failed to traverse are removed
- Reconstruct graph at fixed intervals



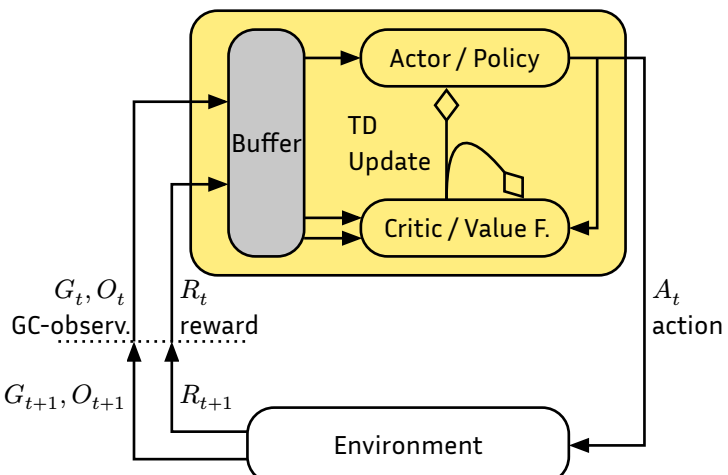
Experiment Setup

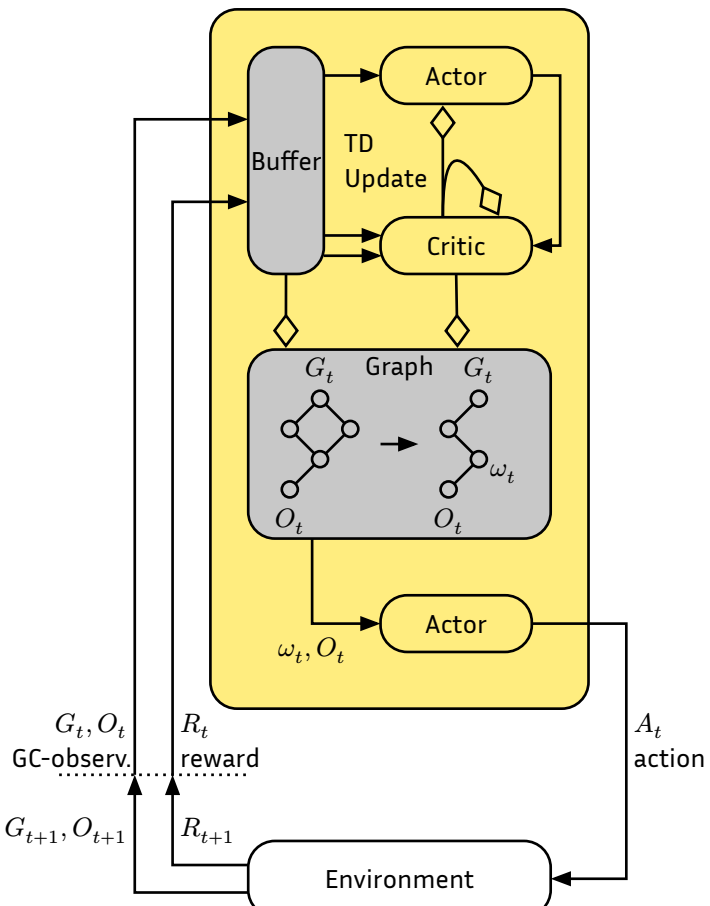
PointEnv Environment





H-DDPG vs HGB-DDPG





The DDPG setting is Goal-Conditioned, but the goal is simple the end-goal.

Challenges

In-Distribution Challenges

Train the agent on one of the 9 environment settings, and measure test-time performance on the same environment.

Out-of-Distribution Challenges

Pretrain the agent on the easiest of the 9 environment settings, and measure test-time performance on a harder environment.

Caveats

We side-step the learned distance measure:

- Assumed access to true distance function

We assume ability to start with a buffer of good quality training data

- uniformly-distributed
- decent amount of rewarded transitions

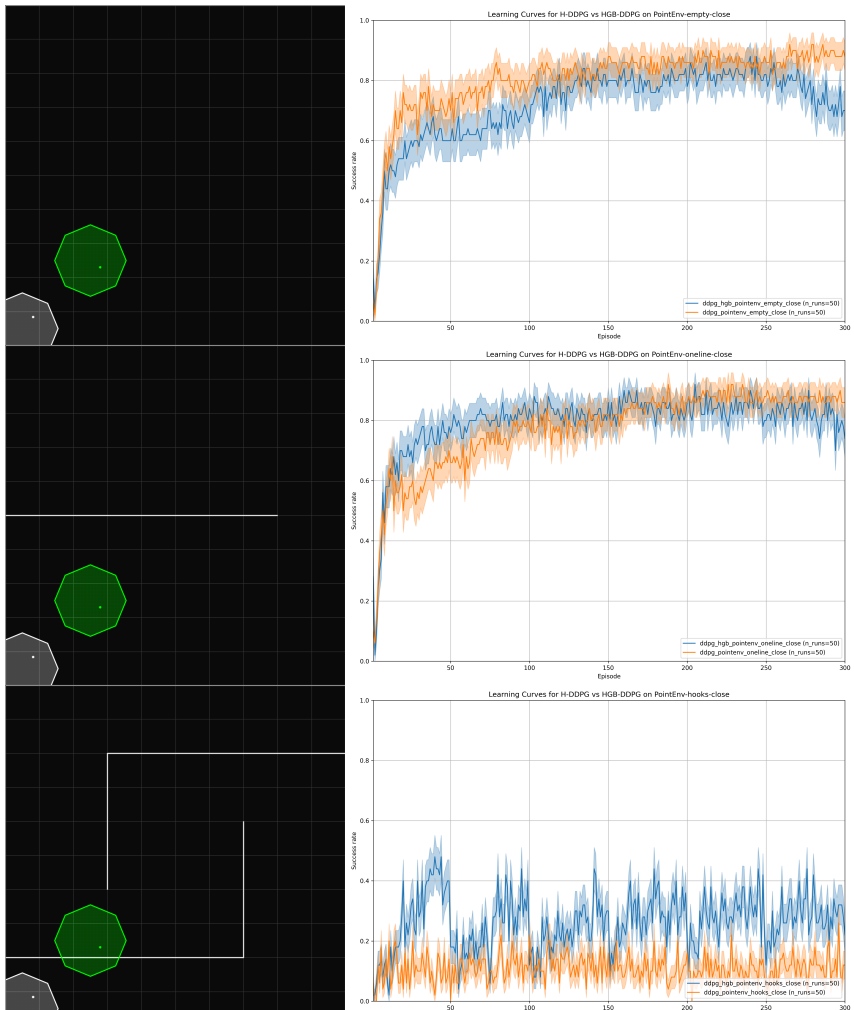
Results

In-Distribution Challenges

- H-DDPG baseline (orange)
- HGB-DDPG algorithm (blue)
- PointEnvs “close”.

We see the expected disadvantage of model-based methods:

- Asymptotic performance is worse
- Overhead of subgoals too high
- Not suitable for short horizon tasks

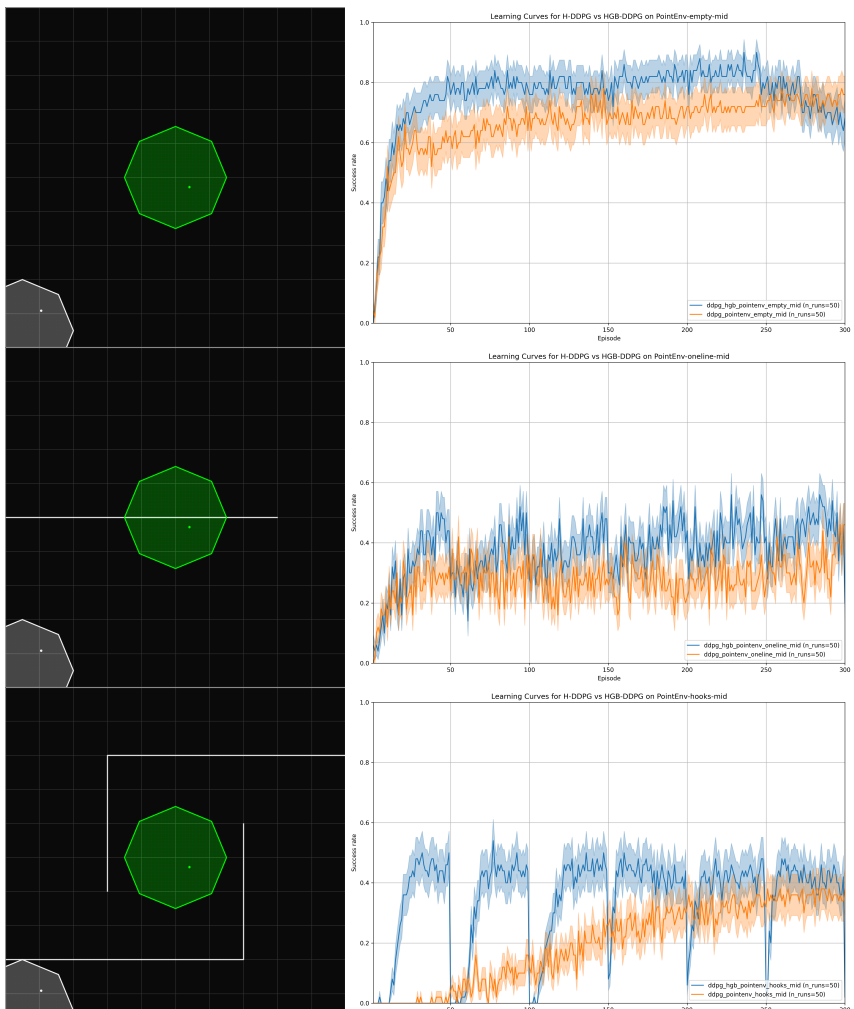


In-Distribution Challenges

- H-DDPG baseline (orange)
- HGB-DDPG algorithm (blue)
- PointEnvs “mid”.

We see the expected disadvantage of model-based methods:

- Asymptotic performance is worse
- Overhead of subgoals too high
- Not suitable for short horizon tasks

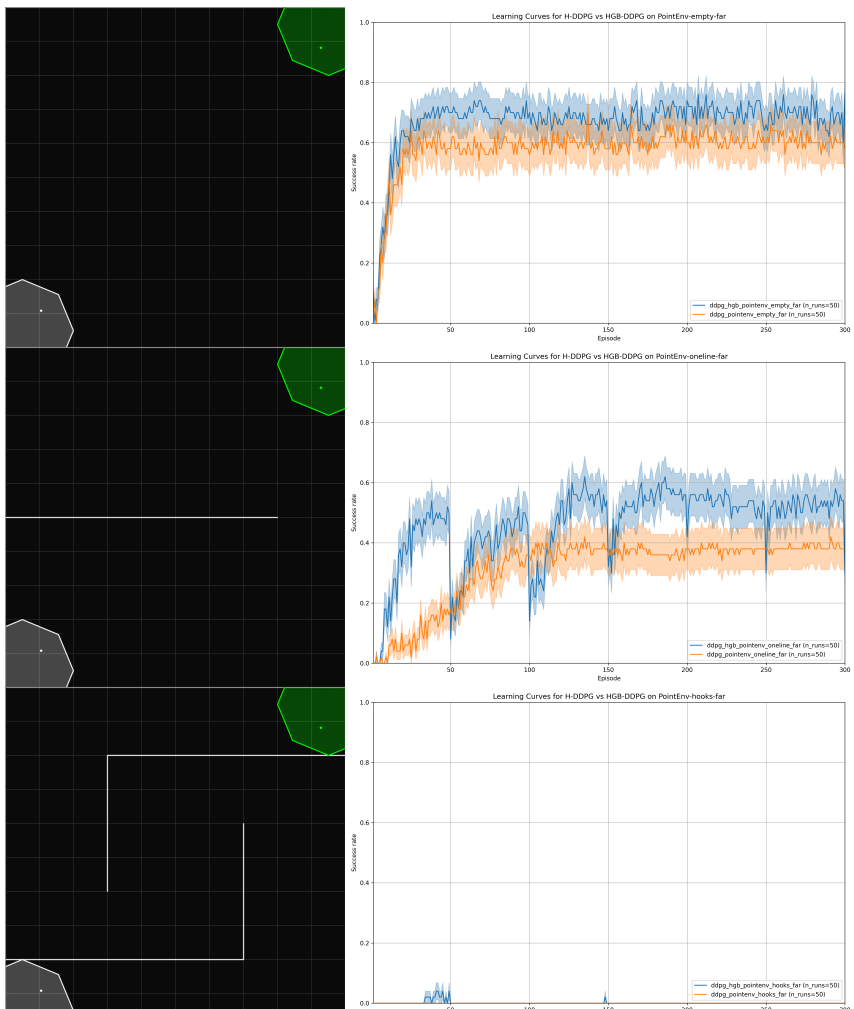


In-Distribution Challenges

- H-DDPG baseline (orange)
- HGB-DDPG algorithm (blue)
- PointEnvs “far”.

We see the expected disadvantage of model-based methods:

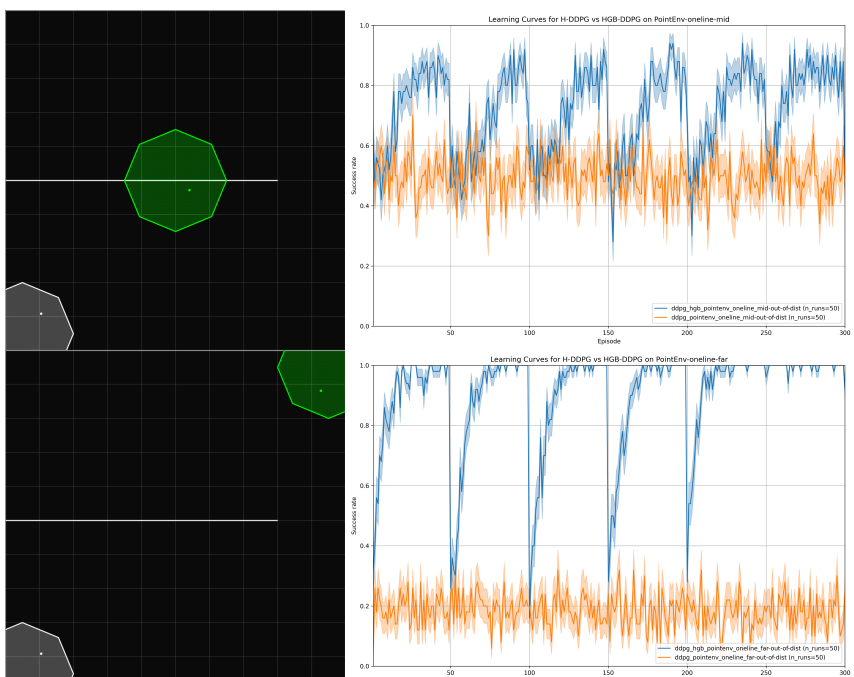
- Asymptotic performance is worse
- Overhead of subgoals too high
- Not suitable for short horizon tasks



Out-of-Distribution Challenges

- H-DDPG baseline (orange)
- HGB-DDPG algorithm (blue)
- Pretrained on PointEnv-Empty-close.

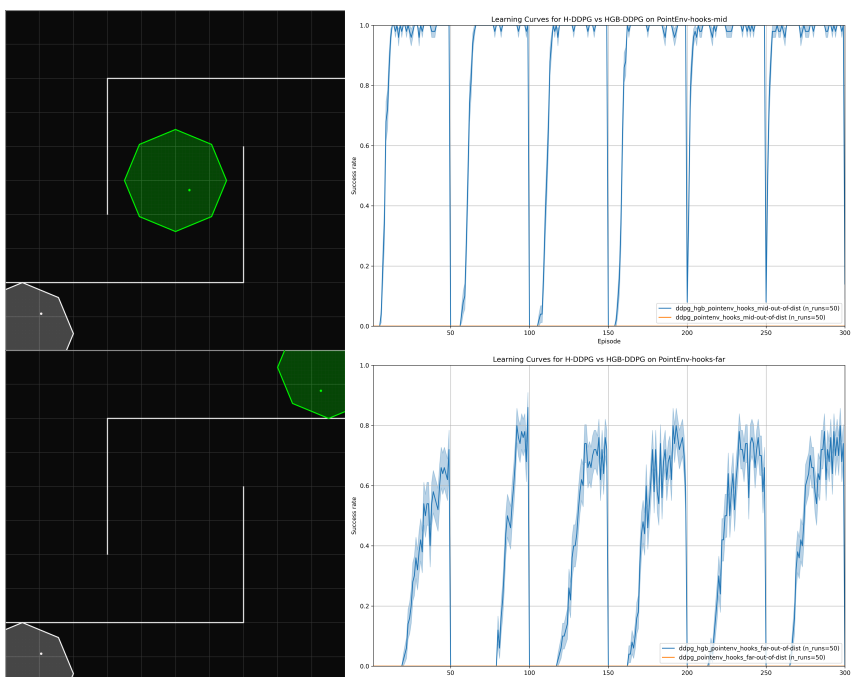
We see the advantages of model-based methods:



Out-of-Distribution Challenges

- H-DDPG baseline (orange)
- HGB-DDPG algorithm (blue)
- Pretrained on PointEnv-Empty-close.

We see the advantages of model-based methods:



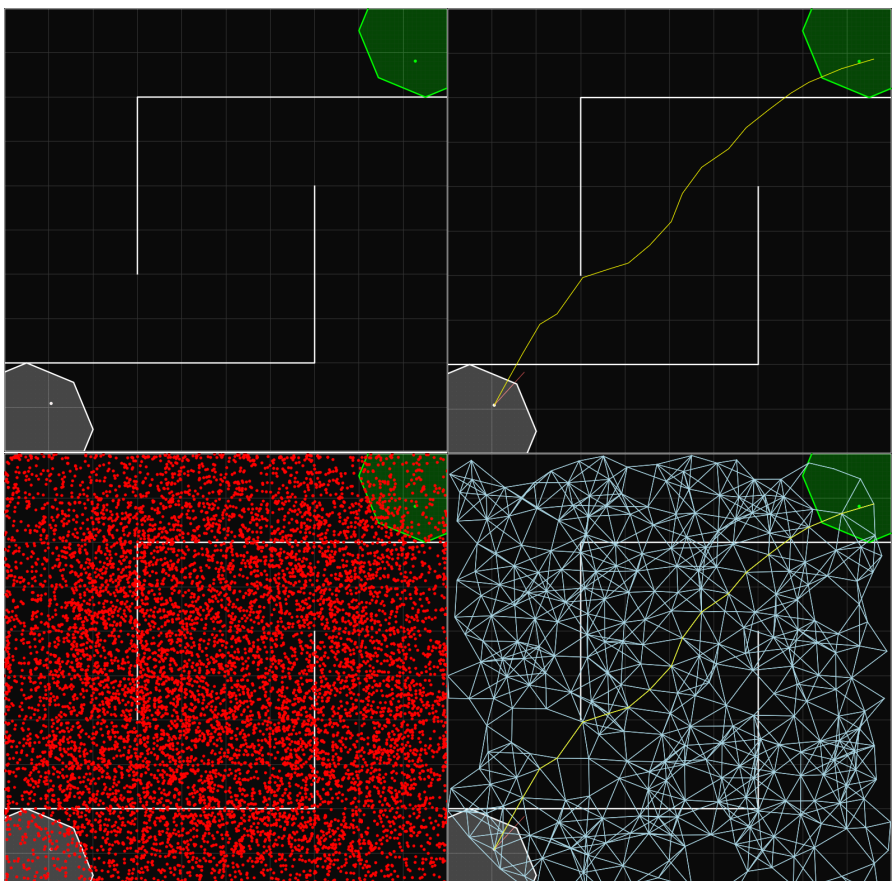
Results on In-Distribution Challenges

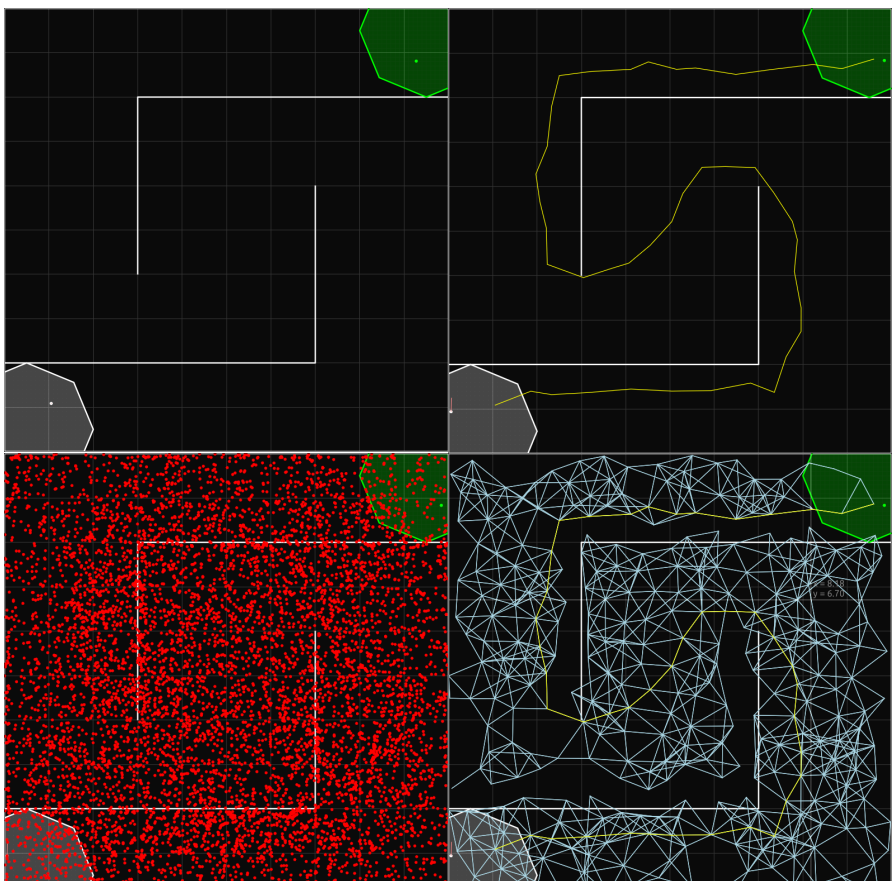
Walls	Distance	H-DDPG (μ)	H-DDPG (SEM)	HGB-DDPG (μ)	HGB-DDPG (SEM)
empty	close	**0.8**	0.05	0.72	0.06
empty	mid	0.67	0.07	**0.76**	0.06
empty	far	0.58	0.07	**0.67**	0.07
oneline	close	0.78	0.06	**0.8**	0.06
oneline	mid	0.28	0.06	**0.38**	0.07
oneline	far	0.31	0.06	**0.46**	0.07
hooks	close	0.11	0.04	**0.26**	0.06
hooks	mid	0.2	0.05	**0.35**	0.06
hooks	far	**0.0**	0.0	**0.0**	0.0

Results on Out-of-Distribution Challenges

Walls	Distance	H-DDPG (μ)	H-DDPG (SEM)	HGB-DDPG (μ)	HGB-DDPG (SEM)
empty	close	**1.0**	0.0	**1.0**	0.0
empty	mid	**1.0**	0.0	**1.0**	0.0
empty	far	0.99	0.01	**1.0**	0.0
oneline	close	**1.0**	0.0	**1.0**	0.0
oneline	mid	0.5	0.07	**0.72**	0.06
oneline	far	0.19	0.05	**0.9**	0.02
hooks	close	0.2	0.06	**0.63**	0.06
hooks	mid	0.0	0.0	**0.85**	0.01
hooks	far	0.0	0.0	**0.33**	0.04

Model Interpretability





Conclusion

Successes

Performance improvement for long-horizon tasks.

The model is interpretable, even editable.

Learning can transfer to new environments (no retraining).

Caveats: Learned Distances

Learned distances are side-stepped by providing the true underlying (euclidean) distance function.

Proof of concept already shown by [2], [3], but implementation exceeded the scope of a MSc thesis.

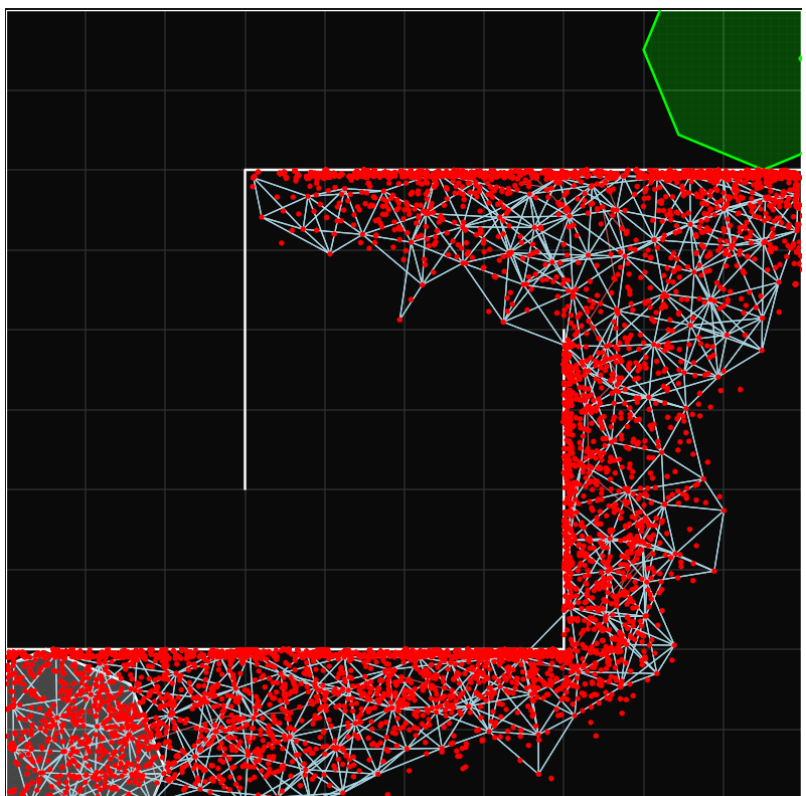
Computationally expensive

Future Research: Exploration

Potential for exploration improvements, e.g:

- Aiming for “frontiers” in the graph
- Keeping statistics on novel nodes found

A solution here would replace our need for pre-filling the replay buffer with uniformly distributed data



Questions

Bibliography

[1] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based Reinforcement Learning: A Survey," *CoRR*, 2020, [Online]. Available: https://arxiv.org/abs/2006.16712

[2] B. Eysenbach, R. Salakhutdinov, and S. Levine, "Search on the Replay Buffer: Bridging Planning and Reinforcement Learning," *CoRR*, 2019, [Online]. Available: http://arxiv.org/abs/1906.05253

[3] M. Laskin, S. Emmons, A. Jain, T. Kurutach, P. Abbeel, and D. Pathak, "Sparse Graphical Memory for Robust Planning," *CoRR*, 2020, [Online]. Available: https://arxiv.org/abs/2003.06417