NEURAL ALL-PAIRS SHORTEST PATH FOR REINFORCEMENT LEARNING

Cristina Pinneri
Max Planck Institute for Intelligent Systems
Tüebingen, Germany
Max Planck ETH Center for Learning Systems
cpinneri@tue.mpg.de

Georg Martius
Max Planck Institute for Intelligent Systems
Tüebingen, Germany

Andreas Krause
ETH Zürich
Switzerland

ABSTRACT

Having an informative and dense reward function is an important requirement to efficiently solve goal-reaching tasks. While the natural reward for such tasks is a binary signal indicating success or failure, providing only a binary reward makes learning very challenging given the sparsity of the feedback. Hence, introducing dense rewards helps to provide smooth gradients. However, these functions are not readily available, and constructing them is difficult, as it often requires a lot of time and domain-specific knowledge, and can unintentionally create spurious local minima. We propose a method that learns *neural all-pairs shortest paths*, used as a distance function to learn a policy for goal-reaching tasks, requiring zero domain-specific knowledge. In particular, our approach includes both a self-supervised signal from the temporal distance between state pairs of an episode, and a metric-based regularizer that leverages the triangle inequality for an additional connectivity information between state triples. This dynamical distance can be either used as a cost function, or reshaped as a reward, and, differently from previous work, is fully self-supervised, compatible with off-policy learning and robust to local minima.

1 INTRODUCTION

Teaching an AI agent to autonomously and efficiently reach goals is still an open problem in the reinforcement learning (RL) community. One of the bottlenecks is correctly specifying a learning signal: an intuitive choice would be a binary reward indicating whether the goal is reached or not, but this kind of signal does not provide enough feedback to the learner, as it would require extensive exploration before receiving any information. As an alternative, hand-crafted reward shaping is often employed on top of the "true" sparse reward - such as the Euclidean distance to the goal (Ng et al., 1999a; Mahmood et al., 2018) - but that might lead to reward hacking, a widely observed phenomenon in which the agent optimizes for the local optima introduced by the shaped reward instead of the real sparse objective, generating unintended behaviors (Pan et al., 2022).

Instead of imposing brittle heuristic shaping, recent research shows that learning *dynamical distances* is a valid alternative (Hartikainen et al., 2019). The purpose of these functions is to learn all-pairs shortest path distances (APSP). One of the first works in this direction was presented in the 90s by Kaelbling (1993) and utilizes a Q-learning agent to construct a goal-conditioned action-value function $d(s, a, g)$, drawing *de facto* the first connection between RL and the APSP problem, based on the Floyd-Warshall algorithm (Floyd, 1962a).

However, Kaelbling (1993) demonstrate the method in a low dimensional tabular setting, where all the goals were known in advance. Recent works (Hartikainen et al., 2019; Jurgenson et al., 2020) propose extensions including function approximation, but are constrained to on-policy learning. We propose a method that learns a dynamical distance function with function approximation, **off-policy**, without any reinforcement signal, employing only a self-supervised signal from the actual number of time steps separating state pairs, and a second loss inspired by the triangle inequality, as in the all-pairs shortest path algorithm.

1.1 Shortest Paths and RL

The problem of learning dynamical distances is closely related to the one of learning value functions in RL. The Q-learning algorithm itself can be seen as a generalization of a single-source shortest path problem (SSSP) for directed graphs, in the context of decision making. The negative Q-function $-Q_\pi(s, \pi(s))$ represents the shortest path distance under the policy π. The dynamic programming (DP) equation underlying Q-learning is the Bellman update (Bellman, 1966), while the one for single-source shortest path is provided by the Dijkstra's algorithm (Dijkstra, 1959) or by other variations like the Bellman-Ford algorithm (Bellman, 1958), depending on the input graph class. However, the nature of these recursive equations can still be traced to the Bellman's optimality principle (Sniedovich, 2006).

Goal-conditioned Q-functions (Schaul et al., 2015) generalize distance learning for a multi-goal setting, as they are defined over pairs of states and desired goals. This is similar to the all-pairs shortest path problem (APSP), which computes the shortest path between any two vertices in a graph. However, the DP equations behind these problems are different. Goal-conditioned Q-learning is typically based on the temporal difference update, where the only bootstrapped value is the estimate of the Q-function. Dedicated algorithms for the solution of the APSP problem are built on different DP principles.

The Floyd-Warshall algorithm (Floyd, 1962b) for APSP, for example, finds shortest paths by iteratively enforcing the triangle inequality between state triples: the shortest path between state s_i and s_k is cast in terms of *relaxations*, such that the distances are initialized by overestimation, and are updated as $d(s_i, s_k) := \min_j [d(s_i, s_j) + d(s_j, s_k)]$. $O(N^3)$ relaxations are needed for the algorithm to converge, where N is the number of vertices in the graph. Kaelbling (1993) propose a RL connection to the Floyd-Warshall algorithm that also includes actions, formalized as $d(s_i, a_i, s_k) := \min_j [d(s_i, a_i, s_j) + \min_{a'} d(s_j, a', s_k)]$. Differently from a temporal difference update, this equation makes use of available estimates of $d(s_i, a_i, s_j)$ as well. Several other works (Kaelbling, 1993; Dhiman et al., 2018; Jurgenson et al., 2020) adapt the Floyd-Warshall algorithm to RL, with results mainly in the tabular setting or for on-policy learning. These methods rely on having accessibility to the ground truth cost between state pairs. Our approach lifts this impractical assumption and learns distances in a self-supervised fashion. Moreover, since we are in a reinforcement learning setting, we can let the agent decide *which* paths to increase depending on their reachability, in a bottom-up approach, rather than initializing all the distances to infinity as in the Floyd-Warshall algorithm.

To summarize, this paper proposes a way to learn dynamical distances and goal-conditioned policies off-policy, and in a fully self-supervised way, employing:

- a novel off-policy correction term for distances learned via temporal regression, and a corrective bootstrapping loss inspired by the Floyd-Warshall algorithm on graphs, adapted to RL.
- a general method to learn dynamical distance functions from scratch, that can be used as part of a goal-conditioned RL algorithm, or as a cost for model-based RL, and that, differently from previous work, successfully deals with local minima (created e.g. by obstacles) without explicitly encoding any type of information in the distance function.

2 Related Work

Dealing with environments with sparse reinforcement signals complicates long-term credit assignment, as it introduces higher variance in Monte Carlo (MC) learning and higher bias in Temporal Difference (TD) updates (Arjona-Medina et al., 2019). Many techniques are focused on accelerating learning by "densifying" the reward signal with auxiliary functions while not affecting the optimal policy of the underlying sparse problem (Grzes & Kudenko, 2008; Randløv & Alstrøm, 1998; Ng et al., 1999b). Alternatively, other kinds of auxiliary reward functions can be used to facilitate exploration according to different strategies as curiosity (Schmidhuber, 1991), prediction error (Pathak et al., 2017), information gain (Houthooft et al., 2016), or predicting state reachability (Savinov et al., 2018). On the other side there are methods like Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) that do not try to solve the exploration problem, but introduce auxiliary rewards based on the distance to the achieved goal rather than to the desired one. As done in HER, we also employ goal-conditioned value functions (Schaul et al., 2015).

Our approach, however, is based on dynamical distances, which are functions expressing the distance between state pairs based on some notion of *functional similarity*. In the literature, dynamical distances have been learned via direct regression using temporal regression Hartikainen et al. (2019), in the form of goal-conditioned policies Ghosh et al. (2019), via Q-learning by relabeling goals Eysenbach et al. (2019); Florensa et al. (2019), or goal-directed Q-learning with negative transition mining Tian et al. (2020). We extend the on-policy temporal regression presented in Hartikainen et al. (2019) to be suited for off-policy learning, while simultaneously learning a goal-conditioned policy that uses the learned dynamical distance as a negative reward. Differently from other methods (Eysenbach et al., 2019; Savinov et al., 2018), we do not explicitly perform graph search. We instead make use of a term that expresses whether two states were ever part of the same trajectory. At training time, we only check if the 'edge' (s, a, s') has ever been visited or not. In practice, we use locality sensitive hashing (Tang et al., 2017), which allows for generalization based on angular distance similarity between states.

3 Background

We consider the problem of an agent attempting to solve a goal-reaching task. The goal reaching Markov Decision Process (MDP) (Schaul et al., 2015; Andrychowicz et al., 2017) is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{G}, r, \gamma, \rho_0, \rho_g)$, where the state, action and goal spaces are indicated by $\mathcal{S}, \mathcal{A}, \mathcal{G} \in \mathbb{R}$. In our case, the goal space $\mathcal{G}$ can be an arbitrary subset of the state space.The initial state s_0 is sampled from the distribution ρ_0, while the goal is sampled from ρ_g. Our aim is to learn a stochastic policy $\pi : \mathcal{S} \times \mathcal{G} \rightarrow \Lambda(\mathcal{A})$ that maximizes the expected discounted return $\mathbb{E}_{\tau \sim \rho_\pi, s_g \sim \rho_g} \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t, s_g) \right]$.

3.1 Hindsight Experience Replay

Let us now consider the particular case in which the reward function is sparse, i.e. the indicator function $r(s_t, a_t, g) = -\mathbb{I}\{s_t \neq g\}$. This assumption complicates the credit assignment problem and becomes a bottleneck for sample efficiency. To overcome this issue, Andrychowicz et al. (2017) introduce Hindsight Experience Replay (HER). The main idea of HER is to relabel past failed trajectories with the indicator function mentioned above, where the goal g is the one that was actually achieved, rather than the desired one.

Although success signals are generally more frequent while using HER, they remain binary, and thus hold little information. In this work we build upon the relabeling framework introduced in HER, but we propose to compute rewards from a more informative learned dynamical distance. Our distance, however, is not tied to the goal sampling procedure presented in HER, and can be used with any relabeling scheme or goal-conditioned RL algorithm as a shaped reward.

3.2 Dynamical Distances

We learn a distance function that predicts the expected number of timesteps along the shortest path between two states. This is learned via a self-supervision signal given by the empirical distance between states of a rollout, as done in Hartikainen et al. (2019), which we extend to an off-policy setting. In the original paper, the distance d_θ^π is associated with a policy π, which gives rise to the training loss:

$$L_d^\pi(\theta) = \frac{1}{2} \mathbb{E}_{\substack{\tau \sim \rho_\pi \\ i \sim [0,T] \\ j \sim [i,T]}} \left[\left(d_\theta^\pi(s_i, s_j) - (j - i) \right)^2 \right], \qquad \texttt{(temporal regression)}$$

where the empirical distance between the states $(j - i)$ is relative to the policy π. The complete dynamical distance learning (DDL) algorithm uses $d_\theta^\pi(s, s_g)$ as a negative reward to optimize the policy π and it is guaranteed to converge to the optimal policy π^*. However, it requires on-policy data collection, which bottlenecks the efficiency of the pipeline. Moreover, it also requires a small amount of goal proposals to be used during on-policy training, selected by a human operator from the last visited goals. In practice, the user is shown a batch of achieved goals and has to select the "best" one, which will be used to train the policy π.

4 Method

4.1 Off-policy Temporal Regression

To overcome the on-policy bottleneck, and achieve better sample efficiency with off-policy data, we propose to condition the dynamical distance on the action value and learn the function $d_\theta(s_i, a_i, s_j)$ from a replay buffer by minimizing the following loss:

$$L_d(\theta) = \mathbb{E}_{\substack{\tau\sim\rho_\pi \\ i\sim[0,T] \\ j\sim[i,T]}} \left[\left(d_\theta(s_i, a_i, s_j) - \min\left[(j-i), d_{\theta_{old}}(s_i, a_i, s_j) + \mathrm{U}[d_{\theta_{old}}(s_i, a_i, s_j)] \right] \right)^2 \right], \quad (1)$$

where θ_{old} are previous distance estimates from a frozen network, updated at a slower frequency for stability reasons, and $\mathrm{U}[\cdot]$ can indicate any measure of uncertainty associated to the network given the input (s_i, a_i, s_j).

Like this, we can train d_θ from suboptimal off-policy trajectories, as the supervision signal is the minimum value between the upper bound of the true distance, given by the empirical distance $(j - i)$, and a frozen estimate of the dynamical distance.

Intuitively, thanks to the generalization capabilities of powerful approximators such as neural networks, $d_{\theta_{old}}$ may provide reasonable lower distance estimates, which can be used as a target in the loss, rather than some temporal overestimate $(j - i)$ coming from suboptimal trajectories at the beginning of training. We explain in further detail our choice of uncertainty measure in section 4.2.

4.2 Uncertainty with Counts

We introduce an uncertainty term in eq. 1 so that the min operator discards distance estimates with high uncertainty, and it starts to consider proposals from the distance network otherwise. Our formulation of the uncertainty term is inspired by a recent work on pessimistic initialisation of Q-functions by Rashid et al. (2020) with count models. We adopt the same model based on locality-sensitive hashing (LSH) (Tang et al., 2017), but we are not bound to any particular counting scheme. We define the count model as $N(s_i, a_i)$ and use it to pessimistically initialise the distance in the following way:

$$\mathrm{U}[d_\theta(s_i, a_i, s_j)] \overset{\text{def}}{=} \frac{C}{(1 + N(s_i, a_i))^M} \qquad \texttt{for the temporal loss} \quad (2)$$

where $M, C > 0$ are hyperparameters. This penalty term polynomially decays with the number of counts, so that the distance gets updated with the off-policy estimate $d_\theta(s_i, a_i, s_j)$ only if that state-action combination has been seen enough times. This pessimistic term is also used to avoid the trivial zero solution of eq. 1.

4.3 Local Connectivity and Triangular Loss

When the goal is not part of the replay buffer and we are in the presence of obstacles, the learned distance underestimates the shortest path to the goal. Let us consider the example in fig. 1. In this case, the initial states distribution ρ_0 samples the starting points from the lower left room. On the other hand, the goals are always sampled from the upper right room. Unless the agent sees a sufficient amount of paths that lead to the goal, it will not be able to learn the correct shortest path distance.

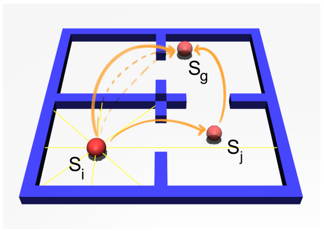
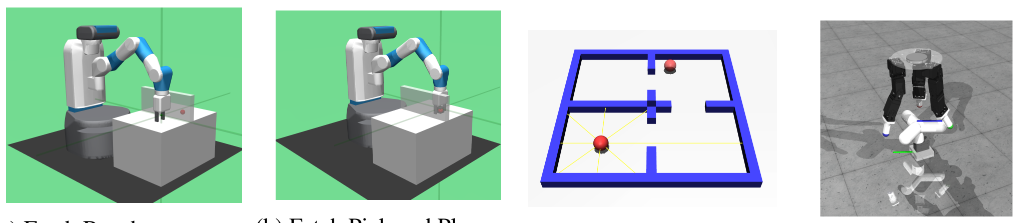


Figure 1: 2D point-mass example. The goal has never been visited. Only the intermediate state is present in the replay buffer. The length of the arrows indicates the distance value between state pairs.

Contrary to the Floyd-Warshall algorithm, where all the distances are initialized with infinity and then "relaxed" with the triangle



(a) Fetch Reach (with and without Wall) (b) Fetch Pick and Place (with and without Wall) (c) Point Maze (d) D'Claw

Figure 2: MuJoCo Environments

inequality, we consider an alternative approach and start with underestimates caused by the off-policy temporal regression, which we artificially increase whenever we sample another point in the buffer that is connected to the goal. In order to express this connectivity information, we employ the count based term already introduced in the previous section to estimate uncertainty. The count term $N(s_i, a_i, s_g)$ then indicates how many times in total the goal s_g was reached from state s_i after taking the first action a_i.

For instance, let us consider the case in which the goal s_g was never reached from state s_i, but there is a path in the buffer $\mathcal{D}$ from an intermediate state s_j to the goal state s_g (fig. 1). In this case, we increase the value of $d_\theta(s_i, a_i, s_g)$ as:

$$d_\theta^\star(s_i, a_i, s_g) = d_\theta(s_i, a_i, s_j) + d_\theta(s_j, a_j, s_g) \tag{3}$$

The distance $d_\theta^\star(s_i, a_i, s_g)$ is then a penalized hypotenuse, that will converge to the true shortest path distance and thus mitigate local minima.

5 Algorithm Summary

The final formulation for the loss comprises the sum of these two terms (from eq. 1 and 3):

$$L_{temporal}(\theta) = \mathop{\mathbb{E}}_{\substack{\tau\sim\mathcal{D}\\ i\sim[0,T]\\ j\sim[i,T]}} \left[\left(d_\theta(s_i, a_i, s_j) - \min[(j-i), d_{\theta_{old}}(s_i, a_i, s_j) + \mathrm{U}[d_{\theta_{old}}]\right)^2 \right] \tag{4a}$$

$$L_{triangular}(\theta) = \mathop{\mathbb{E}}_{\substack{s_g\sim\mathcal{D}_{last}\\ (s_i,a_i)\sim\mu(s_g)\\ (s_j,a_j)\sim\overline{\mu}(s_g)}} \left[\left(d_\theta(s_i, a_i, s_g) - (d_{\theta_{old}}(s_i, a_i, s_j) + d_{\theta_{old}}(s_j, a_j, s_g)) \right)^2 \right] \tag{4b}$$

where $\mu(s_g) = \{(s, a) | (s, a) \in \mathcal{D}_{last} \wedge N(s, a, s_g) = 0\}$ and $\overline{\mu}(s_g) = \mathcal{D} \setminus \mu(s_g)$

where $\mathcal{D}_{last}$ is the replay buffer containing the most recently collected rollouts. Notice that in the triangular loss we can either increase or decrease the value of the hypotenuse. In one case the hypotenuse is penalized, in the other one it is regularized (relaxed, as in Floyd-Warshall) because a better path was found. Moreover, in this loss, s_g always indicates the desired goal of the trajectory which s_i is sampled from. Thus, we are only correcting the paths that should have led to the goal. We underline that this procedure does not search for the (s_j, a_j) tuple minimizing the triangle inequality, it purely samples (s_j, a_j) from the points actually present in the replay buffer $\mathcal{D}$.

6 Experiments

Our method is tested on four environments illustrated in fig. 2:

2D navigation A point mass in two dimensions has to navigate a playground with internal walls to reach a goal. In the setting we present, using the Euclidean goal-distance as a heuristic cost would fail the task because of the presence of local optima created by the walls.

3D reaching task A 7DoF robot arm has to reach a goal. In the Fetch Reach With Wall variation, the robot arm is initialized in one half of the table while the goal is placed on the other half, on the ground, with a wall separating the table in two halves.

Algorithm 1: Proposed Neural-APSP algorithm.

```
Input  : $\mathcal{D}$: empty replay buffer; $\mathcal{D}_{last}$: replay buffer of most recently collected rollouts; $\theta$:
         distance network parameter; $\theta_{old}$: distance target network parameter; $I$: training
         iterations
Output : goal-conditioned policy $\pi$, distance network $d_\theta$
for $i = 0$ to $I-1$                                                    // loop over training iterations
do
    $\mathcal{D}_{last} \leftarrow$ COLLECT DATA WITH POLICY $\pi$
    UPDATE COUNT MODELS with data from $\mathcal{D}_{last}$
    Train TemporalLoss:
        sample episode from buffer $\mathcal{D}$
        sample two indices $i$ and $j$, with $j > i$
        fit distance network with input $(s_i, a_i, s_j)$ and label $\min[j - i, d_{\theta_{old}} + \mathrm{U}]$    // eq. 4a
    UPDATE TARGET NETWORK
    Train TriangularLoss:
        sample $s_i$ from trajectory $\tau$ in $\mathcal{D}_{last}$ and set $s_g$ to desired goal state of $\tau$
        sample $s_j$ from full replay buffer $\mathcal{D}$
        if $N(s_i, a_i, s_g) == 0$ and $N(s_j, a_j, s_g) > 0$ then        // path present from $s_j$ but not $s_i$
            compute new hypotenuse $d^\star(s_i, a_i, s_g)$ using the sampled $s_j$        // eq. 3
        else
            do nothing
        fit distance network with input $(s_i, a_i, s_g)$ and label $d^\star(s_i, a_i, s_g)$        // eq. 4b
    Train Policy $\pi$:
        sample episodes from the replay buffer $\mathcal{D}$
        relabel with HER and reward given by negative distance
        train $\pi$ with SAC on relabeled data
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{last}$
...
```

3D manipulation task A 7DoF robot arm has to fetch, pick and place a box to a target location. The goal can be sampled either above or on the table with 50% probability. In the FETCH PICK AND PLACE WITH WALL, the box is always placed on one half of the table, and the goal in the other, so the the only solution is lifting it.

Claw Manipulation (Ahn et al., 2020) A 9-DOF "claw"-like robot is required to turn a valve to various positions. The state space includes the positions of each joint of each claw (3 joints on 3 claws) and embeds the current angle of the valve in Cartesian coordinate $(\theta \to (\sin\theta, \cos\theta))$. The robot is controlled via joint angle control. The goal space consists only of the claw angle, which is sampled uniformly from the unit circle.

For all the baselines and ablations we use HER (Andrychowicz et al., 2017) while learning a policy with Soft Actor Critic Haarnoja et al. (2018), while for the FETCH PICK AND PLACE environment we use HER combined with DDPG (Lillicrap et al., 2016) due to better empirical performance. All the methods make use of the *future* strategy presented in Andrychowicz et al. (2017), each of them with the best k selected by grid search. The parameters used in the experiments are reported in the appendix. We consider 4 kinds of reward coupled with HER:

Sparse+HER (baseline) Standard binary reward with hindsight relabeling of the buffer with the default SAC hyperparameters.

DDL+HER (baseline) We use the original HER relabeling scheme, and rewards given by the negative temporal loss (on-policy). This method is slightly different from (Hartikainen et al., 2019) as it does not make use of the original "user preferences" to propose goals.

Off-policy DDL+HER (ablation) We augment DDL+HER by introducing the off-policy temporal regression loss term of sec. 4.1.

Ours+HER Our complete method, including the off-policy temporal regression loss (eq. 4a) and the bootstrapped triangular loss (eq. 4b), as described in algorithm 2.

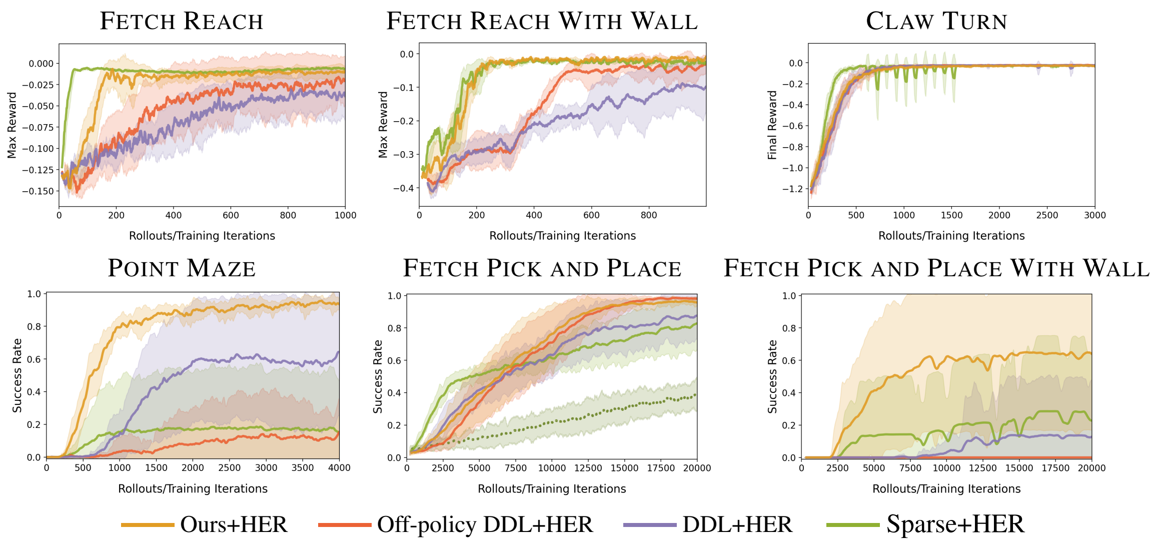


Figure 3: Performance degradation of Sparse+HER. When more exploration is needed or the environment presents non-trivial reachability properties, HER struggles to find a solution. Our method is either comparable to the best baseline/ablation, or better. The quantity presented in the plots is the rolling mean of the maximum reward or the success rate at evaluation time, averaged over 10 seeds. For the FETCH PICK AND PLACE task we augment the goal space to include the end effector position as well. The dotted darker line is the original HER performance without augmentation.

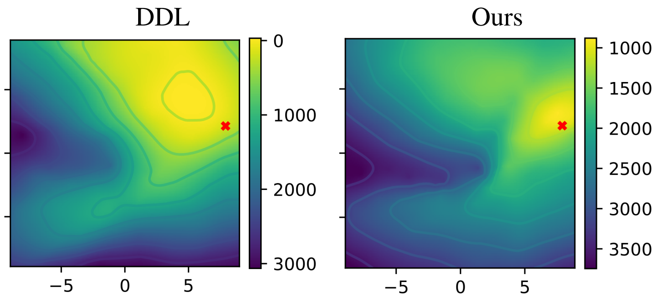


Figure 4: Heatmap of the Q function estimate (negative sign) for the POINT MAZE environment, learned by using DDL+HER (left) and Ours+HER (right).

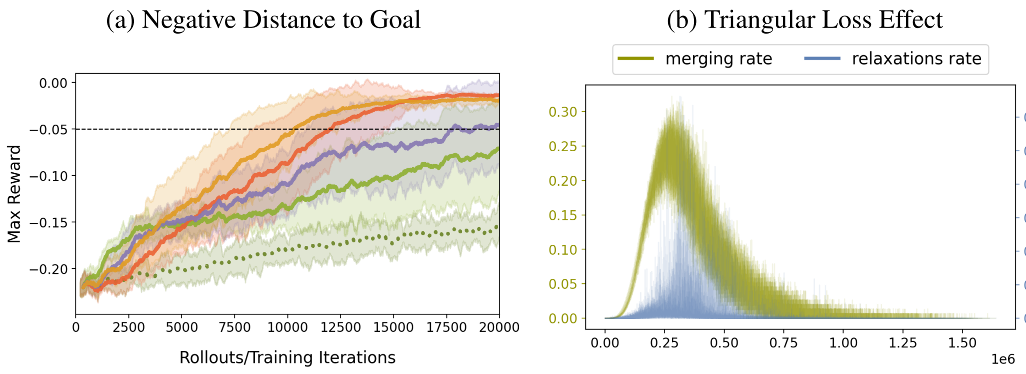


Figure 5: Fetch Pick and Place task. Our method is able to reach the goal at a faster rate as it avoids the local minima of placing the end effector in between the target in the air and the box on the ground. The dashed black line indicates the minimum negative distance required to solve the task. The dotted dark green line is the original HER performance (Andrychowicz et al., 2017) without goal augmentation.

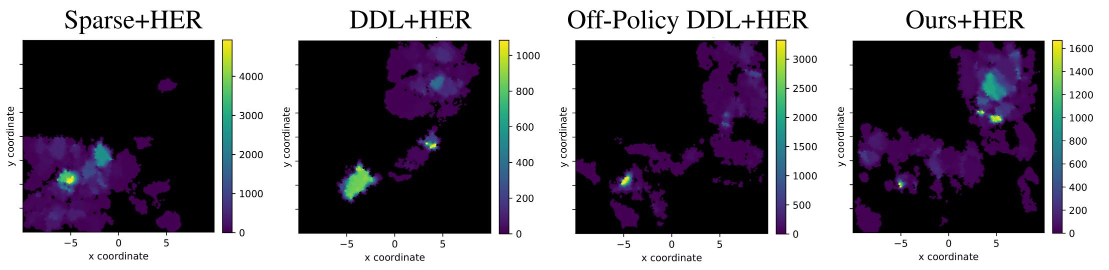


Figure 6: State coverage of POINT MAZE obtained with each method. Sparse+HER is only able to explore the lower room and parts of the adjacent one. All the methods are provided with either $\epsilon = 0.1$ or $\epsilon = 0.3$ exploration, depending on which performs better.

6.1 LOCAL OPTIMA

The results in fig. 3 show how our method is robust even when we consider environments with obstacles or goals that are hard to reach. The off-policy temporal loss can find shortcuts as suggested by the function approximator, and introducing the triangular loss further improves performance In particular, the FETCH PICK AND PLACE task is an eloquent example. In this case we opt to augment the goal space to also consider the end effector position: the goal is to have both box and gripper at the target state. This simple addition already improves the sample efficiency of Sparse+HER by an order of magnitude. However, we can also notice how Sparse+HER quickly reaches a 50% success rate, equivalent to reaching all the goals placed on the table, but cannot generalize so easily to the goals in the air. In fact, given the augmented goal state, it often reaches a suboptimal solution by placing the end effector in between the current box position and its target position. While our method takes more time to catch up, it eventually overtakes both baselines. In fig. 5 we show the effect of the triangular loss for the FETCH PICK AND PLACE task. In particular, we computed the rates at which the distances to the goal get increased (merging rate) or regularized (relaxations rate).

In the FETCH PICK AND PLACE WITH WALL task, it is even more clear that the triangular loss consistently helps avoiding local minima. The robot arm is able to lift the box and bring it to the other side in most cases, without the need to specify an auxiliary reward to avoid the wall. More details in Appendix B.1.

6.2 SAMPLE EFFICIENCY

Being off-policy, our method can make better use of the transitions present in the replay buffer. In fig. 6 we see how, even if all methods are trained with a fixed exploration noise and a max-entropy policy, only our approach is able to consistently explore the goal distribution (top right room). In the specific example of the POINT MAZE, Off-Policy DDL+HER fails to solve the task on its own as it underestimates the distance to the goal. In fig. 4 we represent the Q-function learned by SAC for DDL+HER, and for Ours+HER. The maximum value of the Q-function (its negative value, in the plot) is not centered around the goal for DDL+HER. We presume this might be due to estimation errors induced by the (off-policy) exploration noise.

We also achieve better sample efficiency for the FETCH PICK AND PLACE task. As shown in fig. 5 (a), we manage to get a maximum reward above the threshold after circa 11k rollouts, employing 45% less samples than DDL+HER, and achieving the same performance of Sparse+HER with less than half of the rollouts. In the variant WITH WALL, the efficiency gain is 5x higher.

7 CONCLUSIONS

Learning distances or reward functions in a self-supervised fashion is a generally difficult problem, as it relies on bootstrapped estimates, which can produce strongly biased and high-variance solutions. Solving it, leads to increased sample efficiency during learning which is highly relevant for reinforcement learning applied to real robotic tasks. In addition, learning the distance function does not require any domain knowledge nor expert demonstrations and is thus of general interest for goal-conditioned tasks. The proposed *neural APSP* algorithm shows promising empirical results which indicate that learning these distances, and the corresponding goal-conditioned policies, is not only possible, but also robust to local optima and sample efficient.

8 Acknowledgements

The authors thank Marco Bagatella, Cansu Sancaktar and Anselm Paulus for their useful feedback. This work is funded by the Max Planck ETH Center for Learning Systems and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC number 2064/1 - Project number 390727645.

References

Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on robot learning*, pp. 1300–1313. PMLR, 2020.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02*, 2002.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04*, 2004.

Vikas Dhiman, Shurjo Banerjee, Jeffrey Mark Siskind, and Jason J. Corso. Floyd-warshall reinforcement learning learning from past experiences to reach new goals. *ArXiv*, abs/1809.09318, 2018.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1: 269–271, 1959.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint arXiv:1906.05253*, 2019.

Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019.

Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5:345, 1962a.

Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, jun 1962b. ISSN 0001-0782. doi: 10.1145/367766.368168. URL https://doi.org/10.1145/367766.368168.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal conditioned policies. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Hye9lnCct7.

Marek Grzes and Daniel Kudenko. Plan-based reward shaping for reinforcement learning. In *2008 4th International IEEE Conference Intelligent Systems*, volume 2, pp. 10–22. IEEE, 2008.

Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

Tom Jurgenson, Or Avner, Edward Groshev, and Aviv Tamar. Sub-goal trees a framework for goal-based reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5020–5030. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/jurgenson20a.html.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, 1993.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016. URL http://arxiv.org/abs/1509.02971.

Ashique Rupam Mahmood, Dmytro Korenkevych, Brent Komer, and James Bergstra. Setting up a reinforcement learning task with a real-world robot. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4635–4640, 2018.

A. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999a.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999b.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JYtwGwIL7ye.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pp. 463–471. Citeseer, 1998.

Tabish Rashid, Bei Peng, Wendelin Boehmer, and Shimon Whiteson. Optimistic exploration even with a pessimistic initialisation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1xGP6VYwH.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SygwwGbRW.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

Moshe Sniedovich. Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics*, 35:599–620, 2006.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 2753–2762, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html`.

Stephen Tian, Suraj Nair, Frederik Ebert, Sudeep Dasari, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Model-based visual planning with self-supervised functional distances. *arXiv preprint arXiv:2012.15373*, 2020.