PBCS: Efficient Exploration and Exploitation Using a Synergy between Reinforcement Learning and Motion Planning

Guillaume Matheron, Nicolas Perrin, Olivier Sigaud – guillaume_pub [at] matheron.eu
Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France

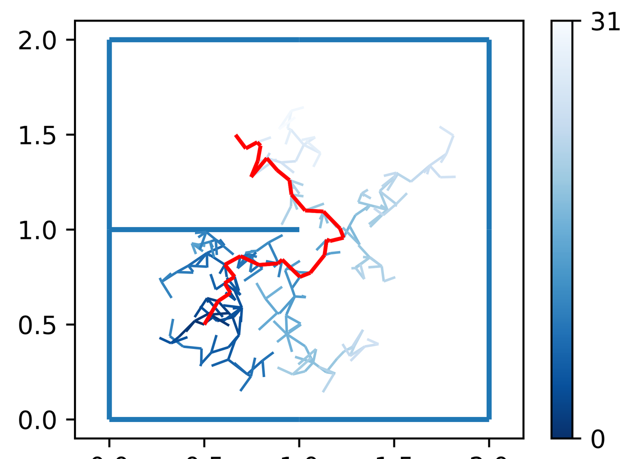
Motion Planning

Pros:
- Is able to explore efficiently without a reward signal.
- Most approaches require a model of the robot.

Cons:
- Learns trajectories, and trajectory following requires a model of the robot.

MP algorithm of choice →RRT [Lavalle, 1998].

We use a variant of RRT which does not require a model (the action is chosen randomly instead of using a heuristic).



In this maze a sparse-reward target at coordinates (.5, 1.5) is easily found. The color range is the depth in the exploration tree.

Algorithm 1: Variant of RRT

In : $s_0 \in S$ the initial state
$d : S \times S \to \mathbb{R}^+$ distance over states
$\text{STEP} : S \times A \to S \times \mathbb{R} \times \mathbb{B}$
RANDOM_STATE, returns a random state from S
RANDOM_ACTION.

Out: transitions $\subseteq S \times A \times \mathbb{R} \times \mathbb{B} \times S$

transitions $= \emptyset$
visited $= \{s_0\}$
while |transitions| $<$ iterations **do**
  $s_{\text{target}} = \text{RANDOM\_STATE}()$
  $s_{\text{near}} = \text{argmin}_{s \in \text{visited}} d(s, s_{\text{target}})$
  $a = \text{RANDOM\_ACTION}()$
  $s', \text{reward} = \text{STEP}(s_{\text{near}}, a)$
  transitions $=$ transitions $\cup\, (s, \text{action}, \text{reward}, s')$
  visited = visited ∪ s'
end

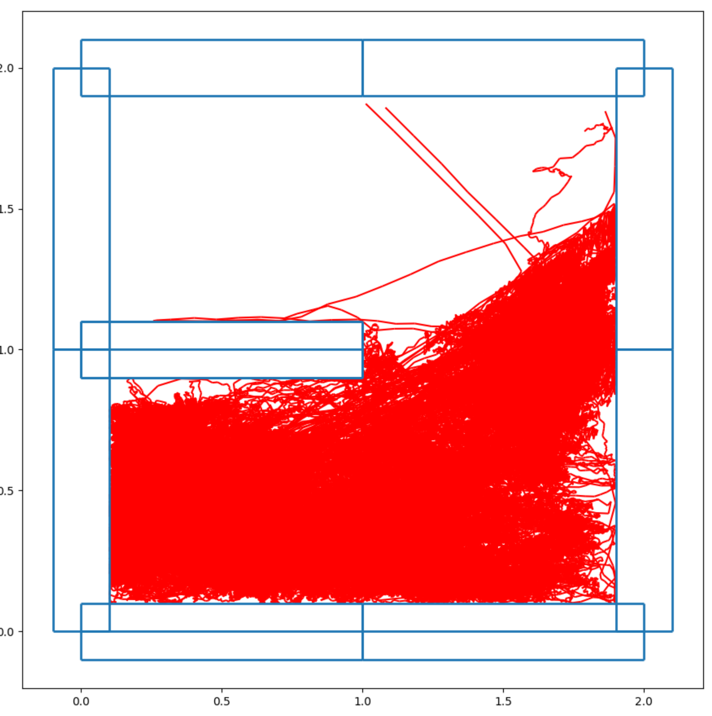
Reinforcement Learning

Pros:
- Does not require a model of the robot.
- Learns a robust controller.

Cons:
- Explores with inefficient random noise when no reward gradient is available.

RL algorithm of choice →DDPG [Lillicrap et al., 2015].

Experience is collected in a replay buffer and the equations to the right are used to update the networks.



Problem: before finding the reward DDPG is only driven by random noise and explores poorly.

DDPG maintains two function approximators:
- an actor π parametrized by Ψ
- a critic Q parametrized by θ that estimates the state-action value function Q^π.

$Q(s_i)$ is made to tend towards $r_i + \gamma Q(s_{i+1}, \pi(s_{i+1}))$.

$$\begin{cases} \forall i, y_i = r_i + \gamma Q_{\theta'}(s_{i+1}, \pi_{\psi'}(s_{i+1})) \\ L_\theta = \sum_i \left[Q_\theta(s_i, a_i) - y_i \right]^2 \end{cases} . \quad (1)$$

$\pi(s_i)$ is changed in order to minimize $Q(s_i, \pi(s_i))$.

$$L_\psi = -\sum_i Q_\theta(s_i, \pi_\psi(s_i)) . \quad (2)$$

Hopefully, this means that $\pi(s_i)$ tends towards $\text{argmax}_a Q(s_i, a)$.

Plan

Motion Planning is used to find a single valid trajectory from the environment start to a rewarded state.
We designed a new algorithm called Ex inspired from EST [Hsu et al., 1997].

Algorithm 2: Ex algorithm

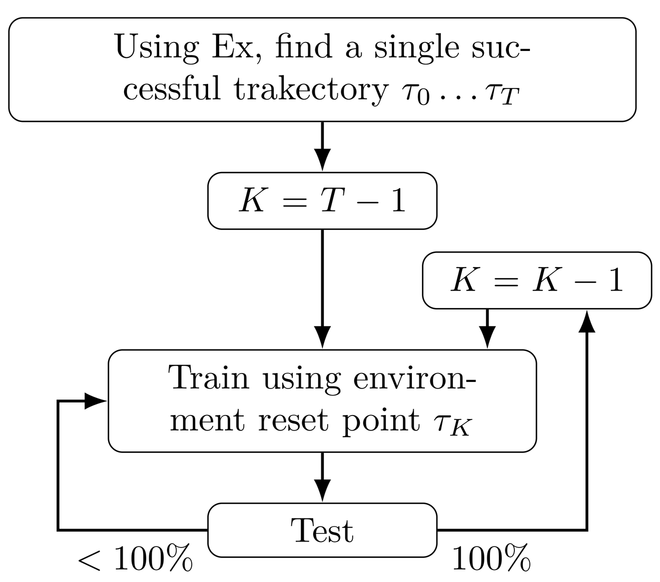
In : $s_0 \in S$ the initial state
step : $S \times A \to S \times \mathbb{R} \times \mathbb{B}$ the step function
iterations $\in \mathbb{N}$
Bin : $S \to \mathbb{N}$ a function that partitions the state-space in bins

Out: T : an exploration tree

Initialize the exploration set T to a single node s_0
$c_{s_0} \leftarrow 0$
while |search tree| $<$ iterations **do**
  $b = \underset{b \in \{\text{Bin}(s), s \in T\}}{\text{argmin}} \sum_{s \in T, \text{Bin}(s) = b} c_s$
  $s = \underset{s \in T, \text{Bin}(s) = b}{\text{argmin}}\ c_s$
  Increment c_s
  a =random_action()
  $s', reward = \text{step}(s, a)$
  $T \leftarrow T \cup \{s'\}$
  If $c_{s'}$ is undefined, then $c_{s'} \leftarrow 0$
end

Backplay

The trajectory obtained in the "Plan" phase is used as a curriculum for training DDPG.
At first, the starting point of the environment is close to the goal, then moved backwards along the curriculum trajectory. This technique is similar to Go-Explore [Ecoffet et al., 2019] and Backplay [Resnick et al., 2018].



Problem: the gradient descent of Eq. (2) may have local minima even when the reward is close and found through random actions! [Matheron et al., 2019]

To counter this, when training becomes stuck the last good policy and K are saved and used as stepping stones for skill chaining.

Chain Skills

The backplay process is wrapped in a skill chaining framework: when backplay fails, it outputs an intermediate point τ_T and a policy π_n that can drive the agent from τ_T to τ_N. Then a new controller is trained to solve the rest recursively.

Algorithm 3: Phase 2 of PBCS

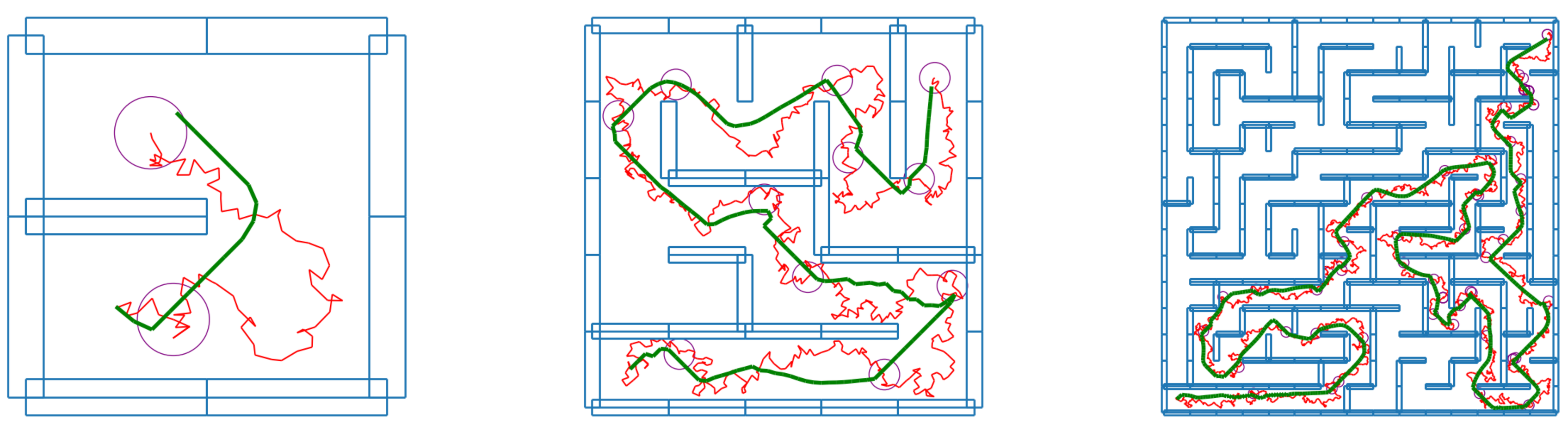
In : $\tau_0 \ldots \tau_N$ the output of the "Plan" phase

Out: $\pi_0 \ldots \pi_n$ a chain of policies with activation sets $A_0 \ldots A_n$

$T = N$
$n = 0$
while $T > 0$ **do**
  $\pi_n, T = \texttt{Backplay}(\tau_0 \ldots \tau_T)$
  A_n = ball of radius ϵ centered around τ_T
  $n = n + 1$
end
Reverse lists $\pi_0 \ldots \pi_n$ and $A_0 \ldots A_n$

The algorithm outputs a list of policies and activation regions that form a multi-policy controller.
To use this controller, we start using π_0 and when the agent reaches a state in A_i, the policy switches to π_i.

Results



Bibliography

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-Explore: a New Approach for Hard-Exploration Problems. *arXiv:1901.10995*, Jan. 2019. URL http://arxiv.org/abs/1901.10995.

D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. volume 3, pages 2719–2726. IEEE, 1997. ISBN 978-0-7803-3612-4. doi: 10.1109/ROBOT.1997.619371. URL http://ieeexplore.ieee.org/document/619371/.

S. M. Lavalle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical report, Iowa State University, 1998.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971*, Sept. 2015. URL http://arxiv.org/abs/1509.02971.

G. Matheron, N. Perrin, and O. Sigaud. The problem with DDPG: understanding failures in deterministic environments with sparse rewards. *arXiv:1911.11679*, Nov. 2019. URL http://arxiv.org/abs/1911.11679.

C. Resnick, R. Raileanu, S. Kapoor, A. Peysakhovich, K. Cho, and J. Bruna. Backplay: "Man muss immer umkehren". *arXiv:1807.06919*, Dec. 2018. URL http://arxiv.org/abs/1807.06919.