RL benchmark for the reaching task

1 Introduction and methods

The Reacher environment is used to benchmark the performance of selected RL algorithms. The Reacher environments are composed of 1 to 6 joints, for example see Figure 1. The RL algorithms tested are: A2C, ACKTR, DDPG, PPO1, PPO2, SAC and TRPO as implemented in the Stable Baselines library using the default parameters. Each agent was trained for 1M time steps. The performance metrics are defined as follows:

1. Train time (min) [1] : Wall time to train for 1M time steps.
2. Success ratio [2] : number of successful episodes / number of reachable episodes (in this case: number of reachable episodes = 100)
   An episode is successful if the distance between the finger tip and the target is less or equal to 0.01. The length of a robot link is 0.1.
3. Average reaching time [3] : sum (number of time steps of all successful episodes) / number of successful episodes
   An episode has a maximum of 100 time steps.

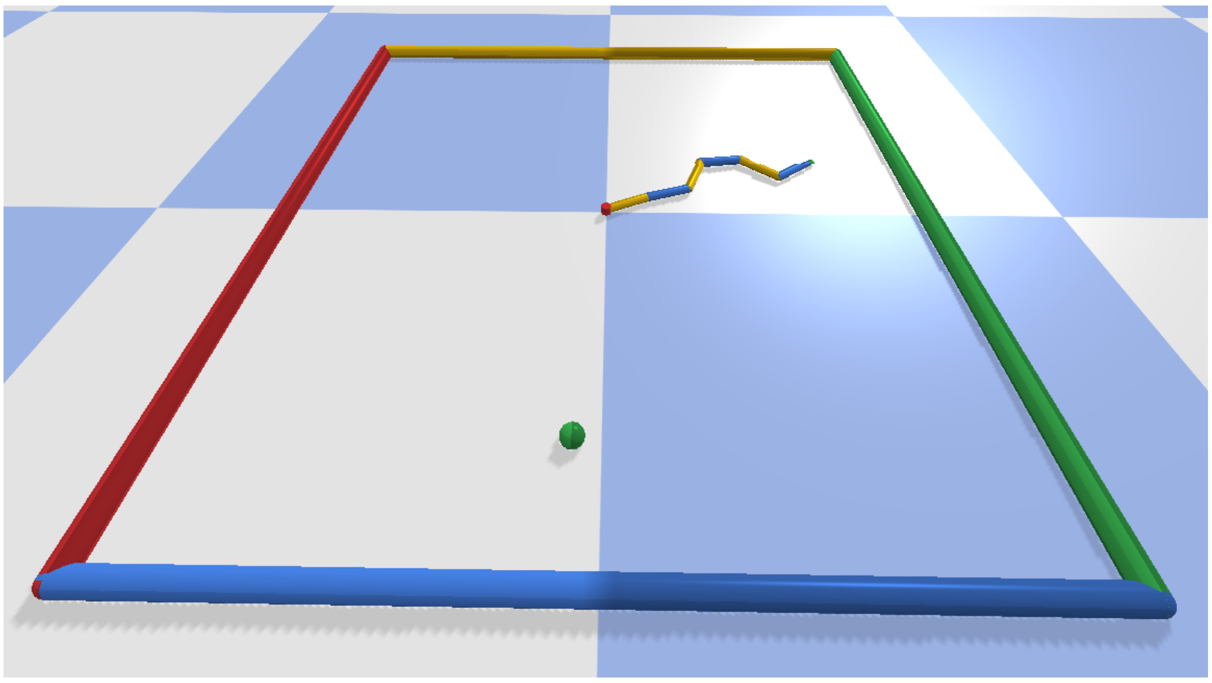
Figure 1: The Reacher environment with 6 DoF.

2 Raw results

Environment	Physics engine	RL algorithm	Train time (min) [1]	Success ratio [2]	Average reach time [3]

Reacher 2D 1 joint	PyBullet	A2C	10.83	0.14	32.36
		ACKTR	8.98	0.16	33.94
		DDPG	31.63	0.07	14
		PPO1	9.01	0.17	17.18
		PPO2	9.57	0.09	8.56
		SAC	49.41	0.13	13.85
		TRPO	7.79	0.12	35.42
		TD3	32.42	0.1	21.9
Reacher 2D 2 joints	PyBullet	A2C	12.13	0.15	39.6
		ACKTR	9.83	0.91	42.73
		DDPG	32.68	0.08	18.75
		PPO1	9.8	0.04	23
		PPO2	10.18	0.09	35.89
		SAC	45.35	0.5	28.59
		TRPO	8.59	0.01	84
		TD3	33.29	0.33	40.64
Reacher 2D 3 joints	PyBullet	A2C	12.34	0.14	41
		ACKTR	10.44	0.59	58.35
		DDPG	33.61	0.12	34.17
		PPO1	10.34	0.09	55.22
		PPO2	10.77	0.12	42.58
		SAC	45.61	0.36	49.03
		TRPO	9.09	0.14	54.93
		TD3	34.13	0.13	36.54
Reacher 2D 4 joints	PyBullet	A2C	12.84	0.06	43.67
		ACKTR	10.93	0.13	49.46
		DDPG	34.51	0.08	57.25
		PPO1	10.96	0.08	26.12
		PPO2	11.25	0.07	45.43
		SAC	46.8	0.27	45.15
		TRPO	9.61	0.13	56.54
		TD3	34.59	0.21	45.38
Reacher 2D 5 joints	PyBullet	A2C	13.43	0.12	49.17
		ACKTR	11.46	0.17	74.65
		DDPG	34.73	0.04	32.5
		PPO1	11.49	0.07	43.43
		PPO2	11.85	0.05	55.8
		SAC	47.11	0.22	49.86
		TRPO	10.16	0.06	62.67
		TD3	36.76	0.18	49.22
Reacher 2D 6 joints	PyBullet	A2C	14.47	0.04	60.25
		ACKTR	12.38	0.14	67.86
		DDPG	35.98	0.11	49.82
		PPO1	12.39	0.05	48
		PPO2	12.61	0.07	30.71
		SAC	46.19	0.14	45.93
		TRPO	10.63	0.06	34.67
		TD3	35.74	0.07	58.14
Jaco 3D 6 joints	ROS / Gazebo	A2C			
		ACKTR			
		DDPG			
		PPO1			
		PPO2			
		SAC			
		TRPO			
		TD3			
Jaco 3D 6 joints	Real world	A2C			

ACKTR
DDPG
PPO1
PPO2
SAC
TRPO
TD3

Complexity to add later:

1. Reach target position and orientation
2. Presence of obstacles
3. Target is moving during operation

3 Performance plots - sorted by algorithm

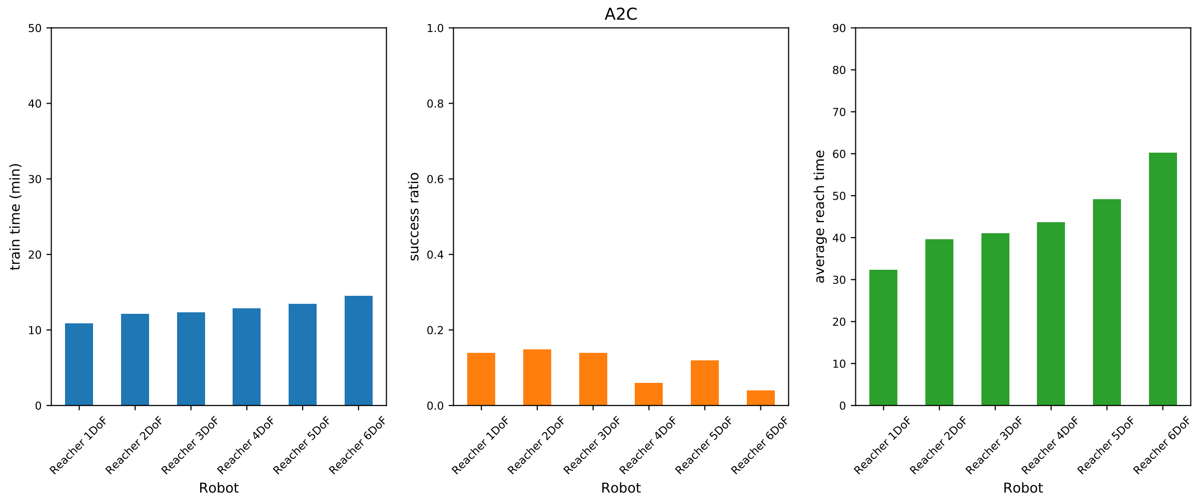

Figure 2: Performance metrics of the A2C algorithm.

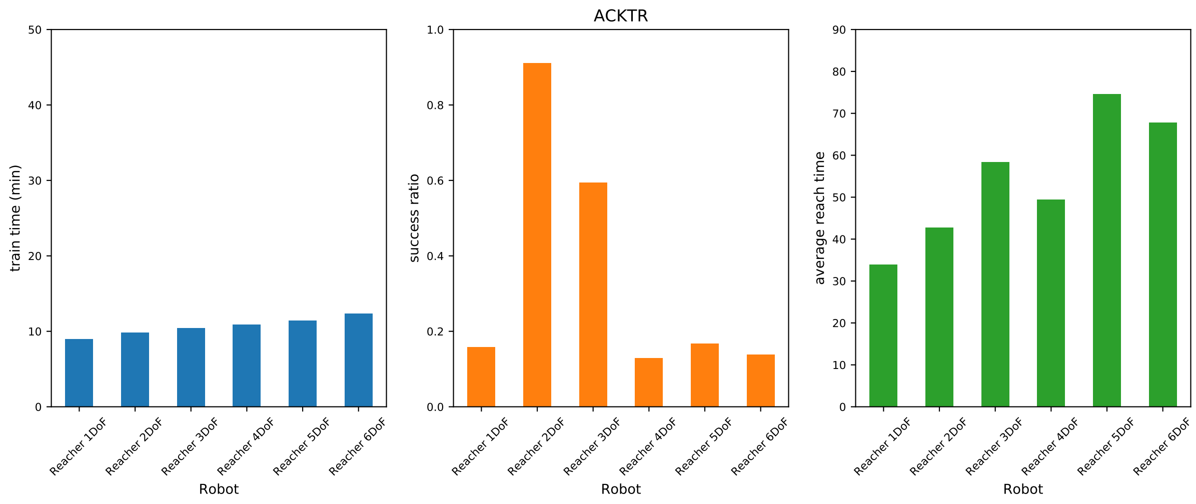

Figure 3: Performance metrics of the ACKTR algorithm.

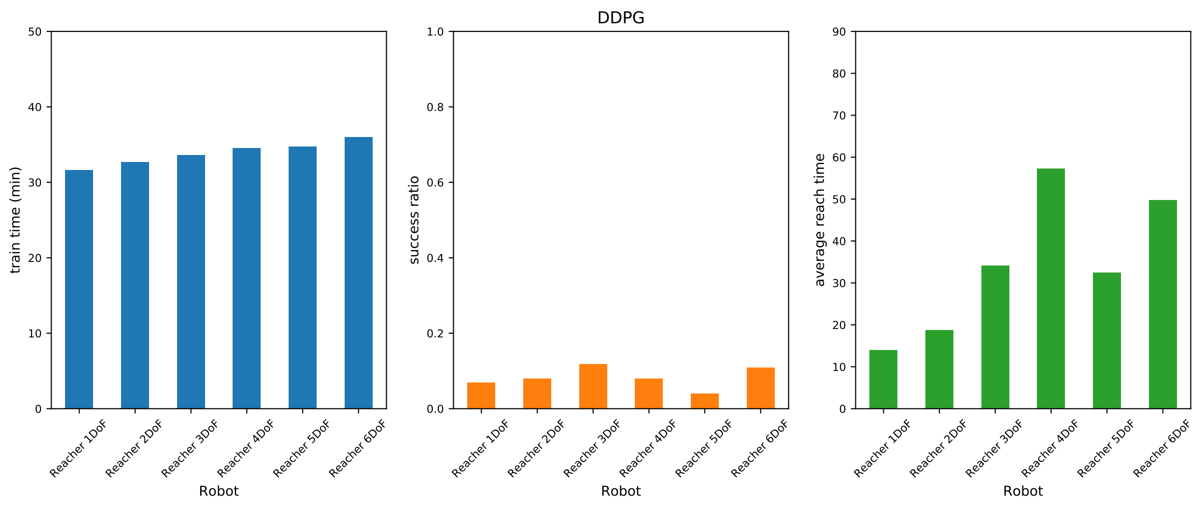

Figure 4: Performance metrics of the DDPG algorithm.

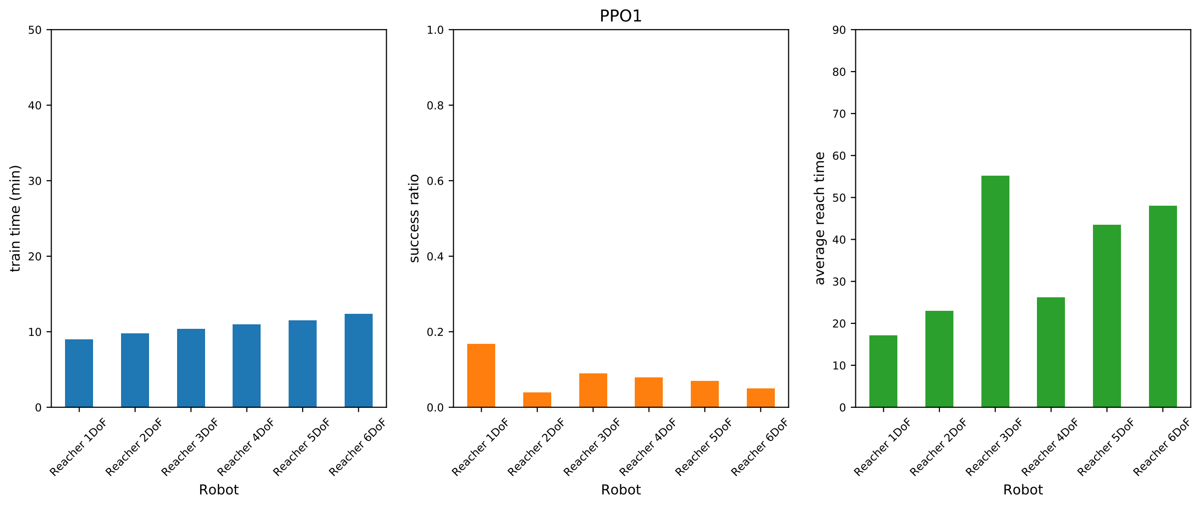

Figure 5: Performance metrics of the PPO1 algorithm.

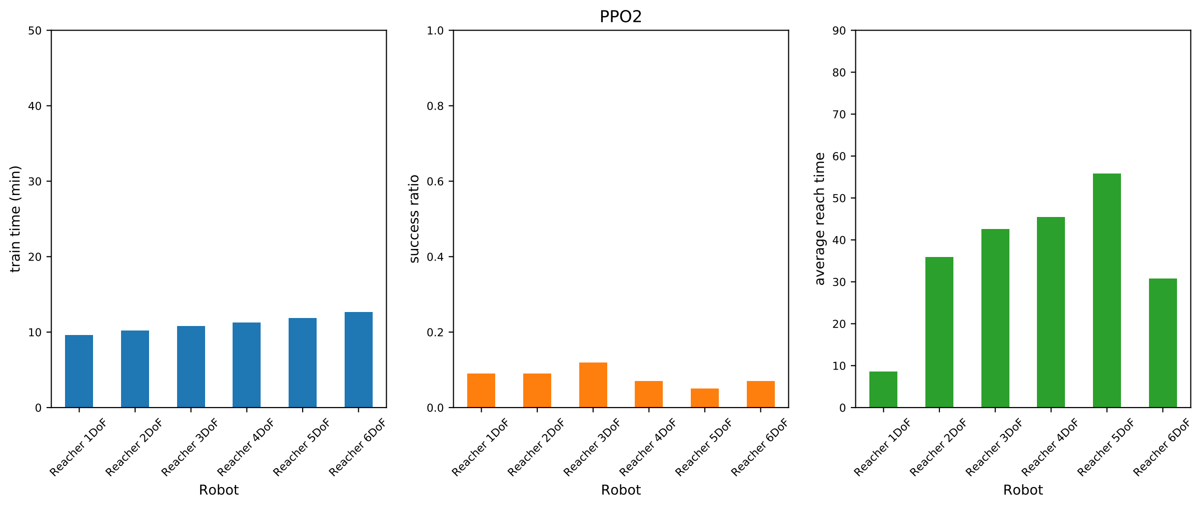

Figure 6: Performance metrics of the PPO2 algorithm.

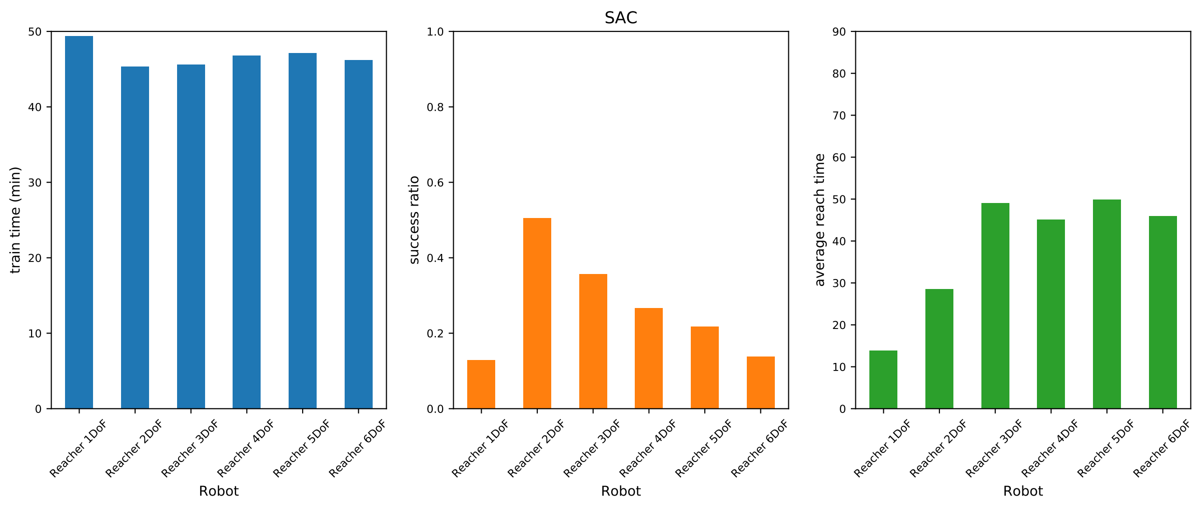

Figure 7: Performance metrics of the SAC algorithm.

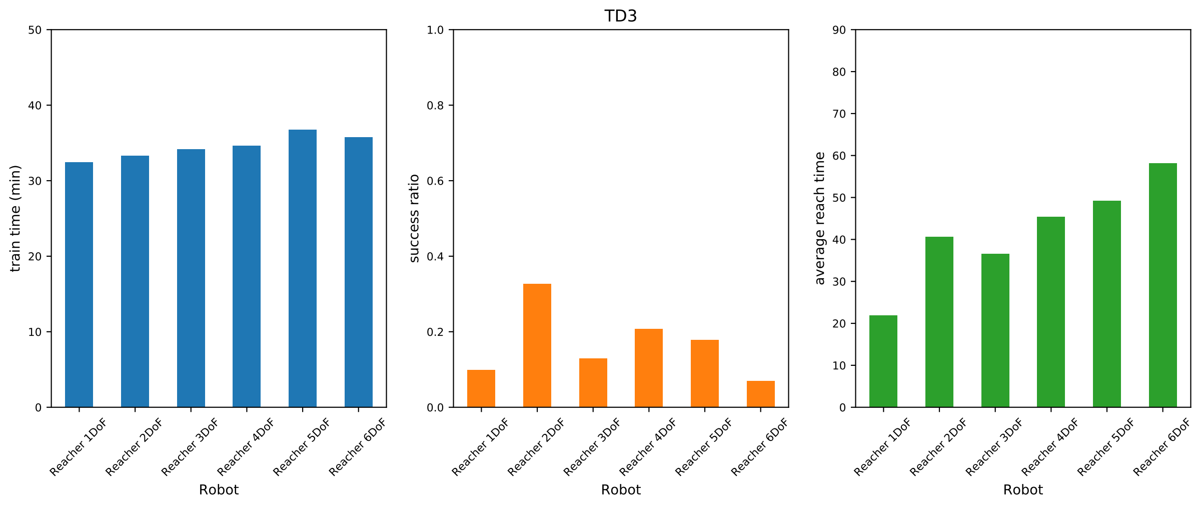

Figure 8: Performance metrics of the TD3 algorithm.

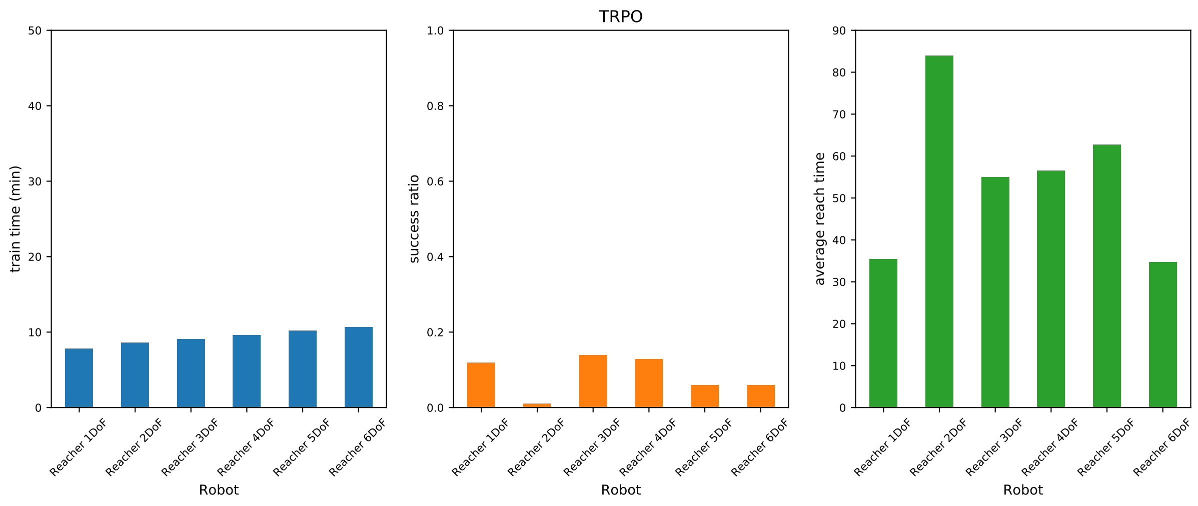

Figure 9: Performance metrics of the TRPO algorithm.

4 Performance plots - sorted by number of joints

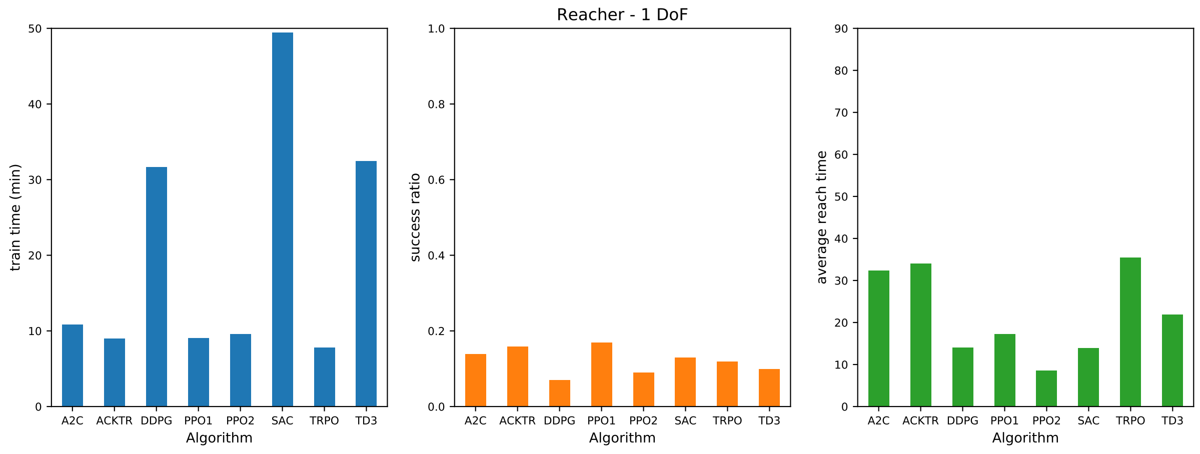

Figure 10: Performance metrics of the Reacher 1 DoF robot.

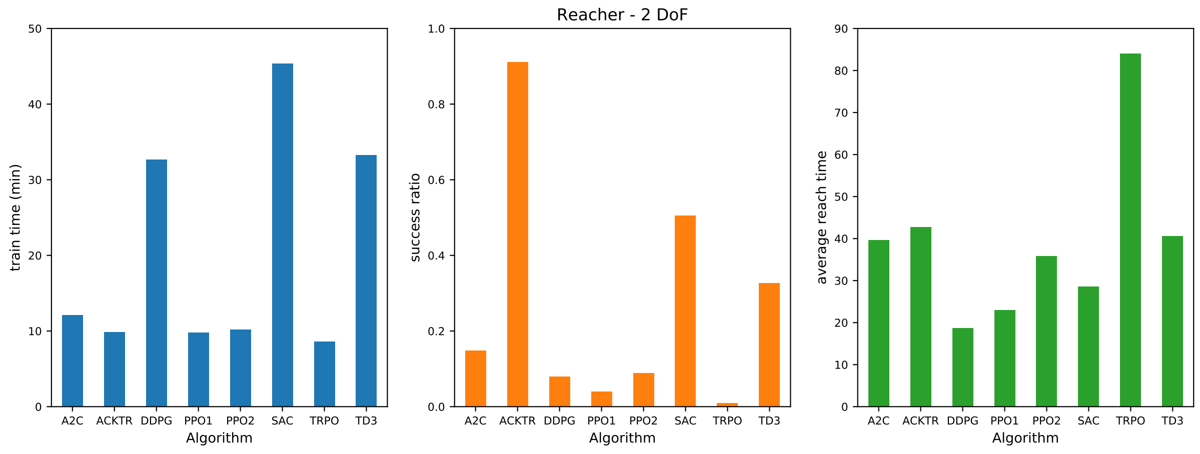

Figure 11: Performance metrics of the Reacher 2 DoF robot.

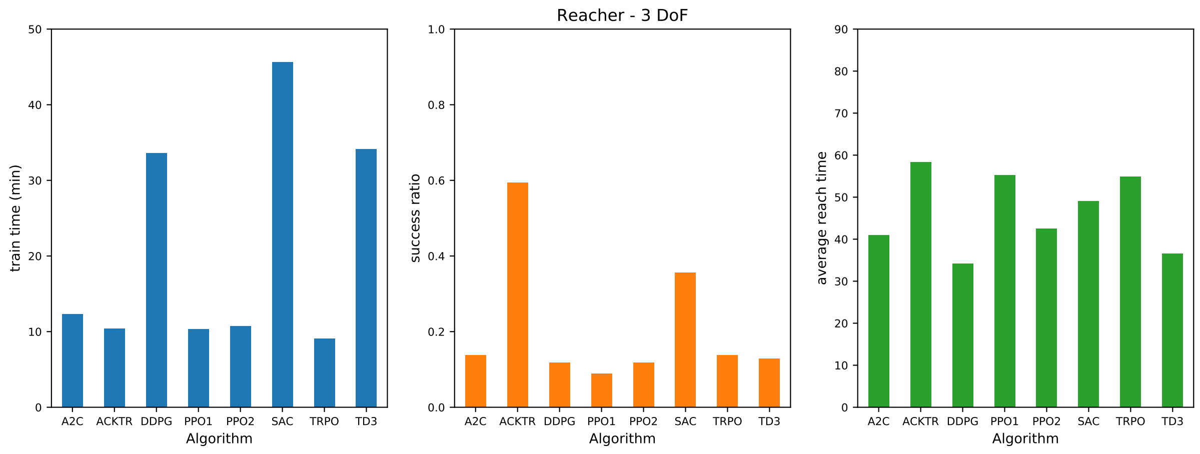

Figure 12: Performance metrics of the Reacher 3 DoF robot.

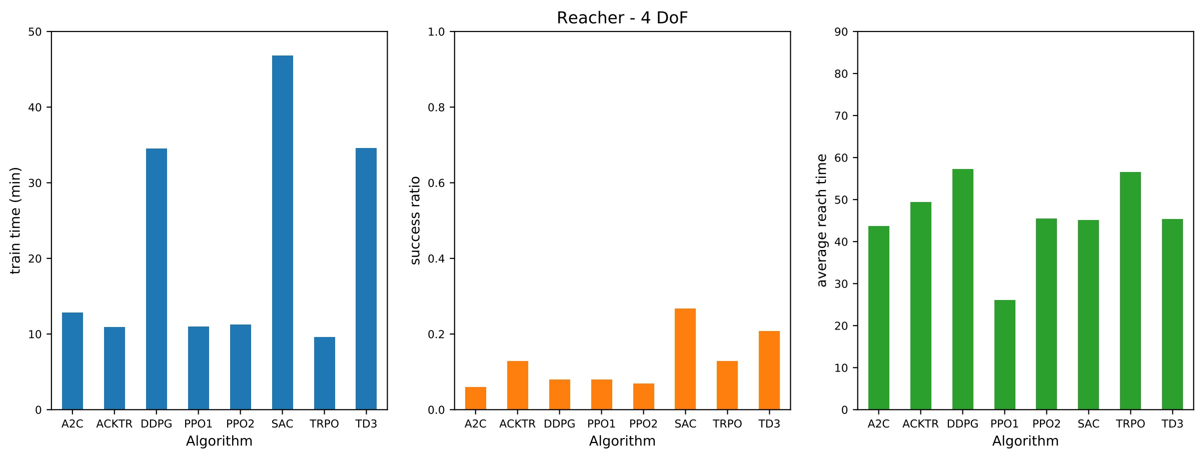

Figure 13: Performance metrics of the Reacher 4 DoF robot.

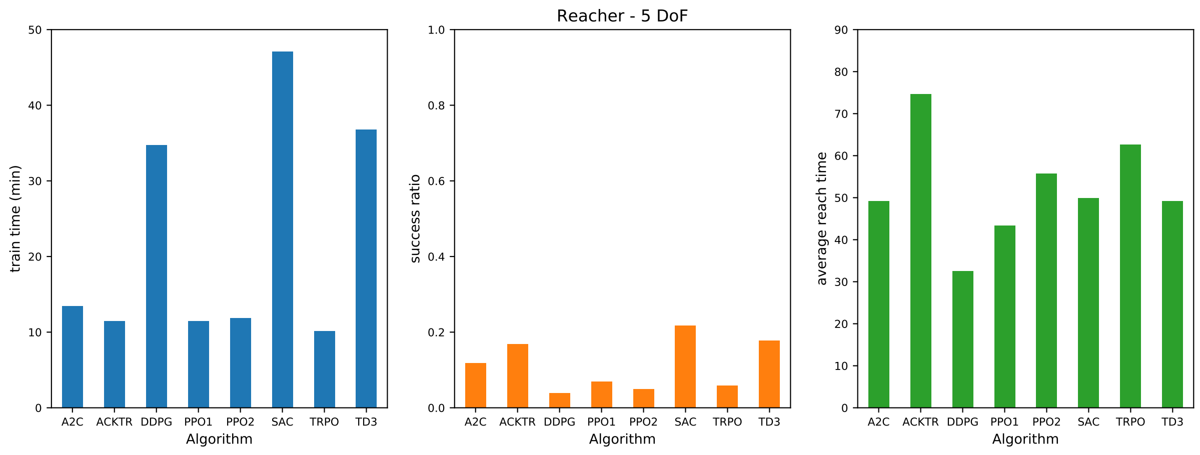

Figure 14: Performance metrics of the Reacher 5 DoF robot.

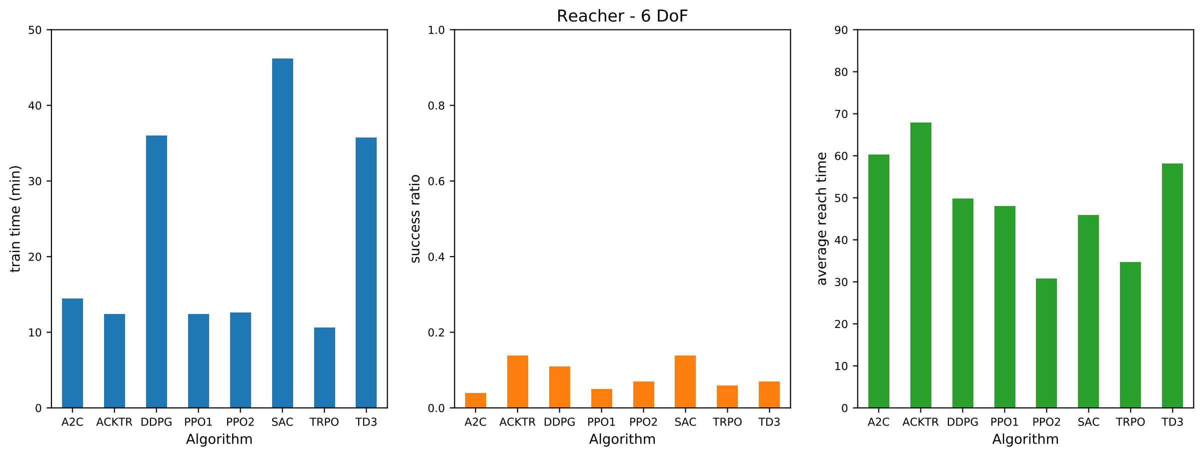

Figure 15: Performance metrics of the Reacher 6 DoF robot.

5 Findings summary

1. The training time increases as the environment complexity increases (i.e. robot with a higher number of joints), which is expected.

2. SAC and DDPG are the least time-efficient algorithms for the Reacher environment.
3. SAC achieved the higher success ratio.
4. In general, the algorithms achieved a very low success ratio. As indicated in the Stable Baselines documentation, it is recommended to normalise the observations, actions and rewards and to tune the algorithm's hyperparameters to increase the performance. Also, regarding the performance evaluation of the agents, it is necessary to initialise the training with various random seeds and take the average performance in order to obtain a more meaningful comparison.
5. A robot with more joints will generally take more time steps to reach the target, as expected.
6. The best agent was trained with ACKTR for the 2DoF Reacher environment (0.91 success ratio).

The next steps will consists in:

1. normalising the observations, actions and rewards.
2. carrying out hyperparameter tuning.
3. initialising the training with various random seeds and average performance over these trained environments.