UPF progress report

Anders Jonsson

DELTA meeting
4 October 2018

Overview

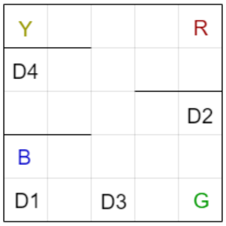
Evaluation of decomposition strategies for lifelong reinforcement learning

Improving width-based planning with compact policies

Summary

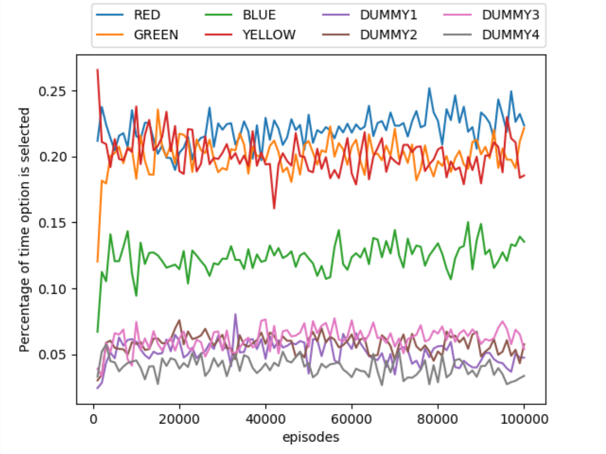
- Master's thesis of Marc Alvinyà
- Aim: evaluate a set of options and identify options that can be pruned
- Idea: adapt action elimination to the hierarchical setting [Even-Dar et al. 2006]

Setting

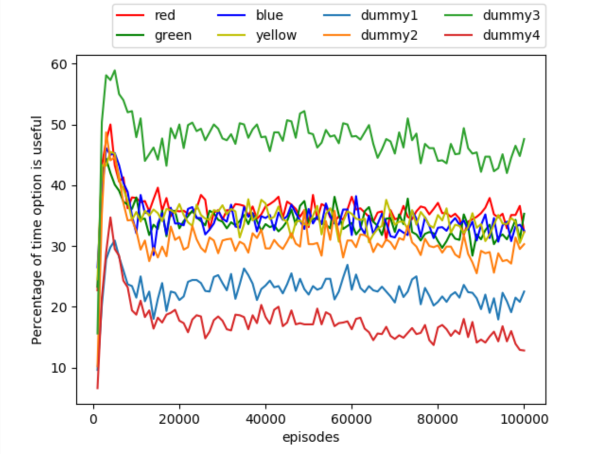


- Fixed set of options in a simple domain
- Each option terminates in a given partial state
- Two evaluation criteria of an option o in state s
  1. Q-value: $Q(s, o)$
  2. Estimated reward + value: $\widehat{R}(s, o) + V(s')$

Q-value


RED
GREEN
BLUE
YELLOW
DUMMY1
DUMMY2
DUMMY3
DUMMY4
Percentage of time option is selected
0.25
0.20
0.15
0.10
0.05
0
20000
40000
60000
80000
100000
episodes

Estimated reward + value



Overview

Evaluation of decomposition strategies for lifelong reinforcement learning

Improving width-based planning with compact policies

Summary

- Paper at the ICML / IJCAI / AAMAS 2018 Workshop on Planning and Learning (PAL)
- Joint work with Miquel Junyent and Vicenç Gómez
- Aim: explore the combination of width-based planning with deep reinforcement learning

Planning and Learning in sequential decision making

Planning Community

- Action model is known and often deterministic
- Goal-directed behaviors
- Heuristic search, Width-based planning, SAT, etc

Machine Learning Community

- Action model is unknown and stochastic
- Maximize expected cumulative reward
- Deep Reinforcement Learning

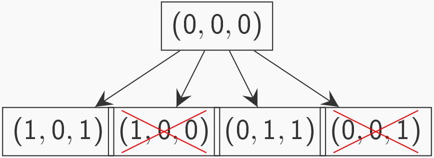
Challenge: Exploit strengths of both approaches

Iterated Width (IW) (Lipovetzky & Geffner 2012)

- A blind (breadth-first) search exploration method for planning
- States factored into features $\phi(s)$
- IW(i): Prunes states that are not **novel for width** i

Novelty

A state s is **novel for width** i if at least one n-tuple of features $\phi(s)$ appears for the first time during search, with $n \leq i$



- **Complexity exponential in i, but independent of $|\mathcal{S}|$**

Different planning approaches

Width-based planning in Atari

- RAM-based features, IW + greedy actions (Lipovetzky+ 2015)
- Pixel-based features, Rollout IW (Bandres+ 2018)

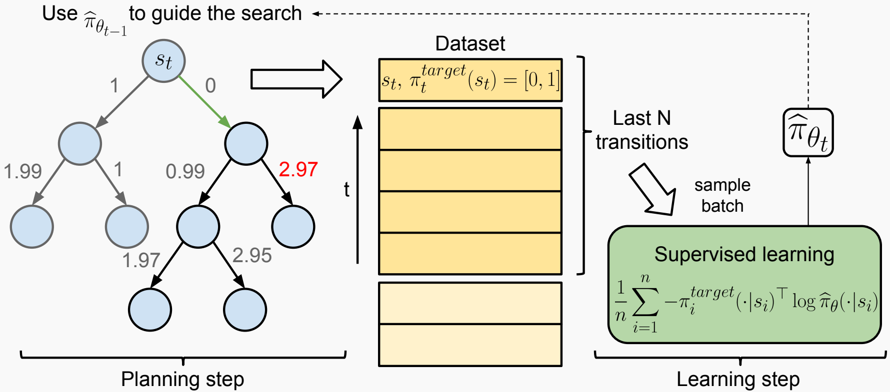
Planning in deep reinforcement learning

- Monte-Carlo tree search
- Integrate planning and learning: AlphaZero (Silver+ 2017), Offline MCTS planning (Guo+ 2014)
- Random exploration, ignores state structure

Our work: Policy-guided Iterated Width (PIW)

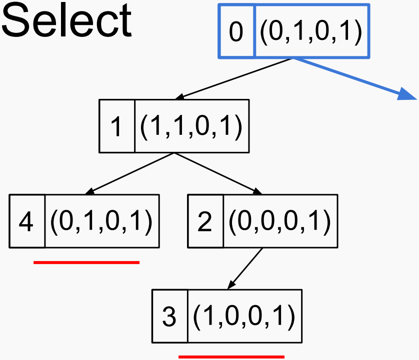
- A novel, structured, exploration strategy
- Imitation learning using IW as the teacher

Policy-guided Iterated Width (PIW)



PIW: planning step

Select



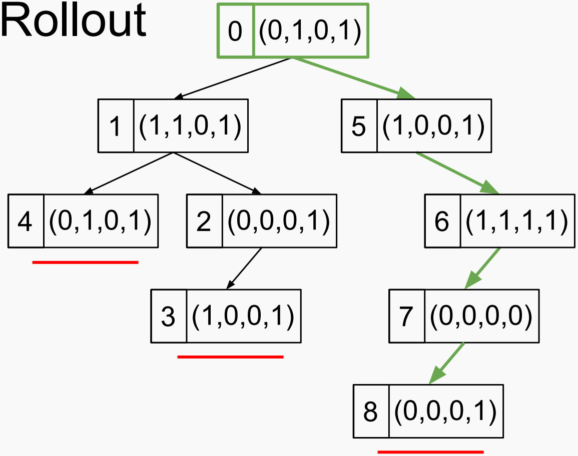
Novelty table

feat.	value	depth
f_1	0	0
	1	1
f_2	0	2
	1	0
f_3	0	0
	1	
f_4	0	
	1	0

Step by step demonstration: http://bit.ly/**PIW-plan**

PIW: planning step

Rollout



Novelty table

feat.	value	depth
f_1	0	0
	1	1
f_2	0	**1**
	1	0
f_3	0	0
	1	**2**
f_4	0	**3**
	1	0

Step by step demonstration: http://bit.ly/**PIW-plan**

PIW: learning step

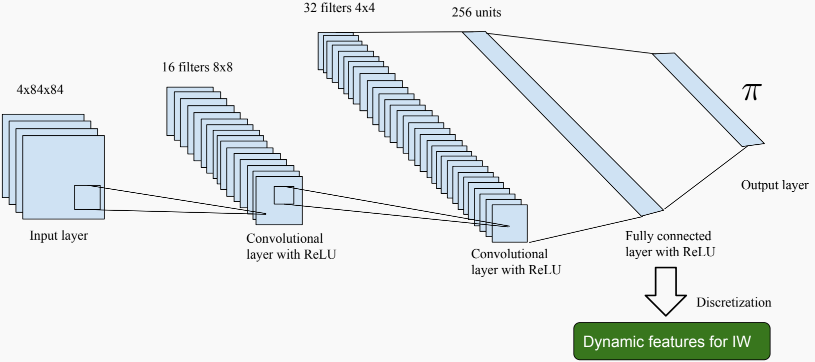
- *Supervised learning*: π^{target} extracted from the plan
- Sample minibatch from the dataset and minimize

$$-\frac{1}{n}\sum_{i=1}^{n}\pi_i^{target}(\cdot|s_i)^\top \log \widehat{\pi}_\theta(\cdot|s_i)$$

- Policy update similar to AlphaZero
- Easily parallelized: More than one actor performing a lookahead, with shared NN parameters.

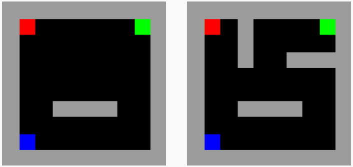
Dynamic features

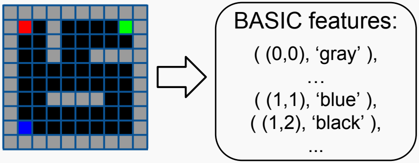
- Automatic and fast feature extraction
- Binary discretization: motivated by the use of ReLUs



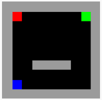
Simple key-door environments

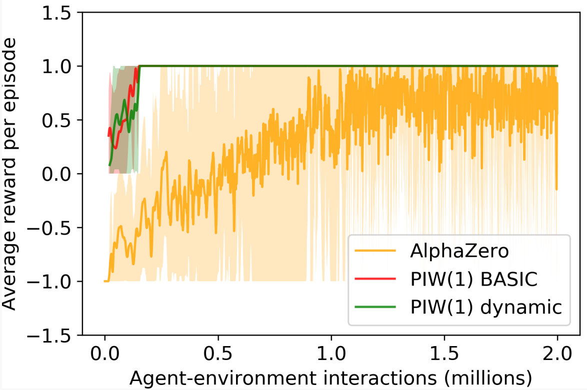
- 10×10 gridworld environment
- Sparse reward, episode terminates:
  - when the agent picks the key and reaches the door ($r = +1$)
  - when hitting a wall ($r = -1$)
  - after 200 steps ($r = 0$)
- The agent starts each episode at position (1,1)





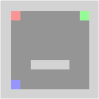
Simple environments: lookahead

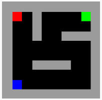


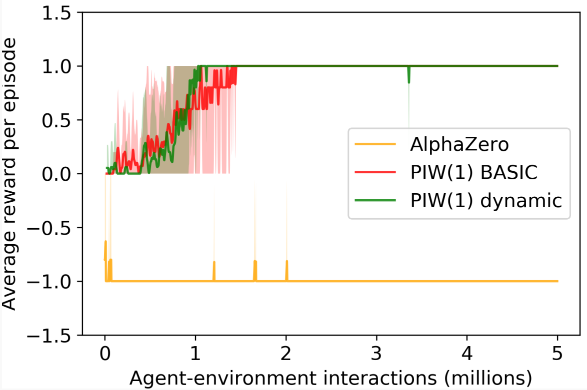


- PIW(1) outperforms AlphaZero
- Similar performance using BASIC and dynamic features

Simple environments: lookahead

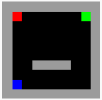


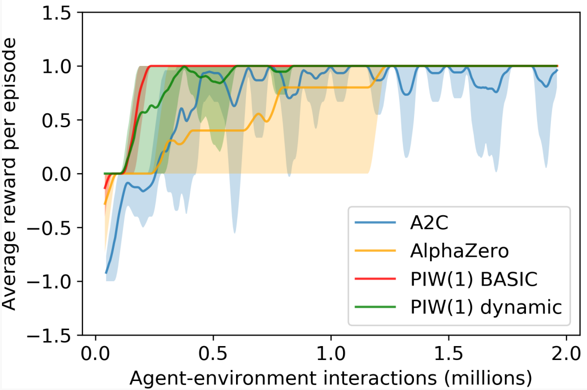




- AlphaZero is not able to achieve the goal in a more complex environment

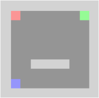
Simple environments: compact policy

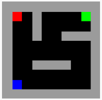


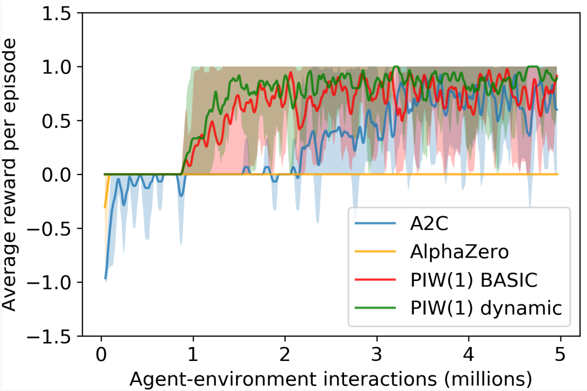


- The compact policy learned by PIW(1) performs better than A2C
- We evaluated it every 20k frames

Simple environments: compact policy







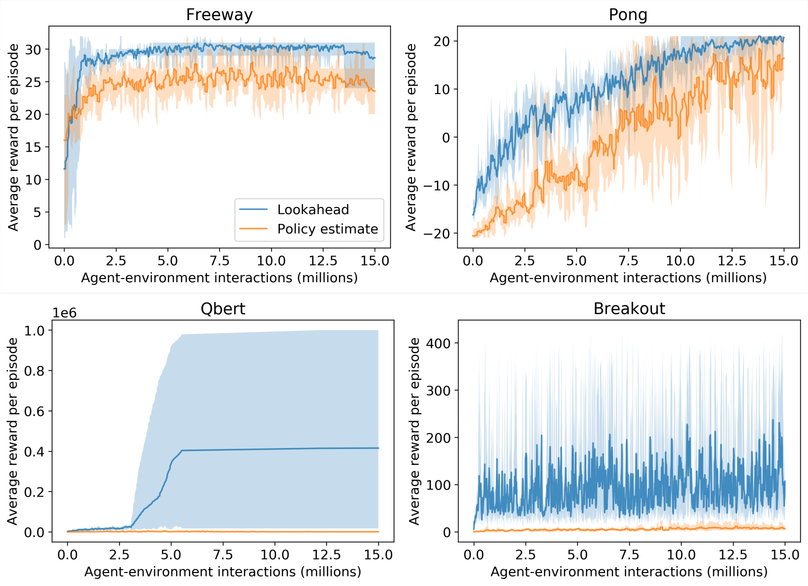
- The policy also outperforms A2C in this setting
- Both algorithms require more steps, and result in a more noisy performance

Results in Atari 2600: lookahead

Game	IW RAM (#1500)	RIW BPROST (0.5s)	RIW BPROST (32s)	PIW dynamic (#100)
Breakout	384.0	82.4	36.0	107.1
Freeway	31.0	2.8	12.6	28.65
Pong	21.0	-7.4	17.6	20.7
Qbert	3,705.0	3,375.0	8,390.0	415,271.5

- PIW outperforms Rollout IW
- Comparable results to IW with RAM
- Few nodes per iteration: #100 $\sim$ 1s (vs #1500/32s)

Results in Atari 2600



Conclusions

Contributions

- PIW: Improves IW planning *reinforcing* promising paths
- Dynamic features: no need for hand-crafted features

- Simple sparse-reward environment:
  - PIW (lookahead) outperfoms AlphaZero
  - PIW (policy estimate) better than A2C
- Atari:
  - Better than Rollout IW with BPROST features
  - Comparable to IW with RAM features (less nodes!)

Results in Atari 2600: compact policy

Game	Human	DQN	A3C	A3C+	PIW
Breakout	31.8	259.40	432.42	473.93	6.9
Freeway	29.6	30.12	0.00	30.48	23.55
Pong	9.3	19.17	20.84	20.75	16.38
Qbert	13,455.0	7,094.91	19,175.72	19,257.55	570.5

- The policy estimate learned by PIW is not able to achieve state-of-the-art RL performance
- Far less training frames: 15M transitions include all generated trees
- Frameskip of 15 instead of 4 as in DQN or A3C