Universität Basel

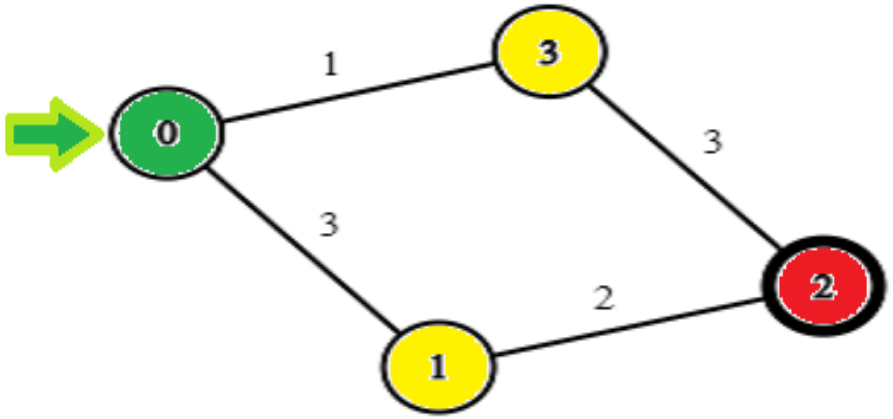
Deep Reinforcement Learning for Online Planner Portfolios

Bachelor Thesis

Tim Goppelsroeder <t.goppelsroeder@stud.unibas.ch>
Department of Mathematics & Computer Science
University of Basel
14.02.2023

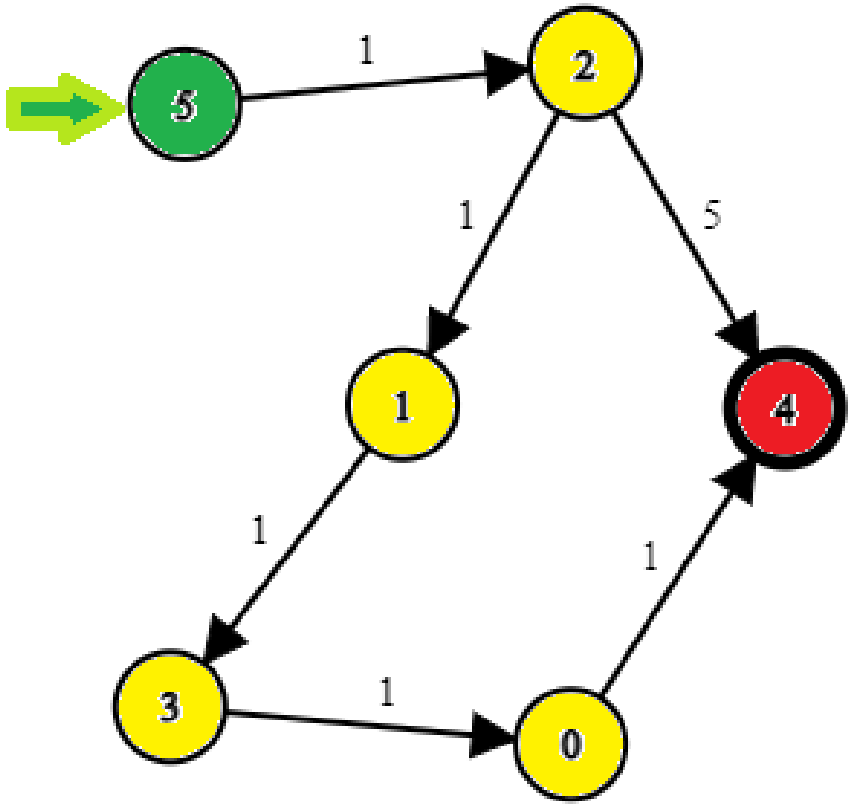
Automated Planning & State Spaces

- Find action sequence leading from initial state to goal state
- Let our state space be $\boldsymbol{S} = (S, A, cost, T, s_0, S_*)$
- Our objective is to find a sequence of actions $(a_1, a_2, \ldots, a_n)$ where we start at s_0 and end at $s \in S_*$ with $a_i \in A$



Optimal Planning

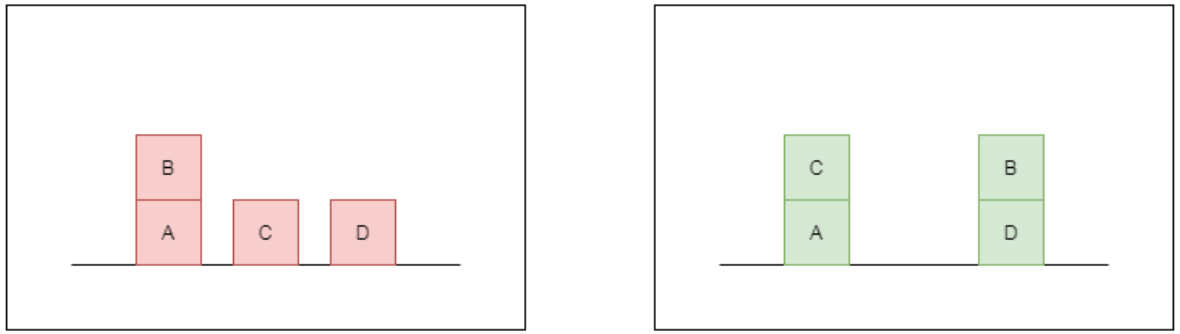
Optimal Planning is concerned with finding a sequence of actions with minimal cost



Planning Formalisms

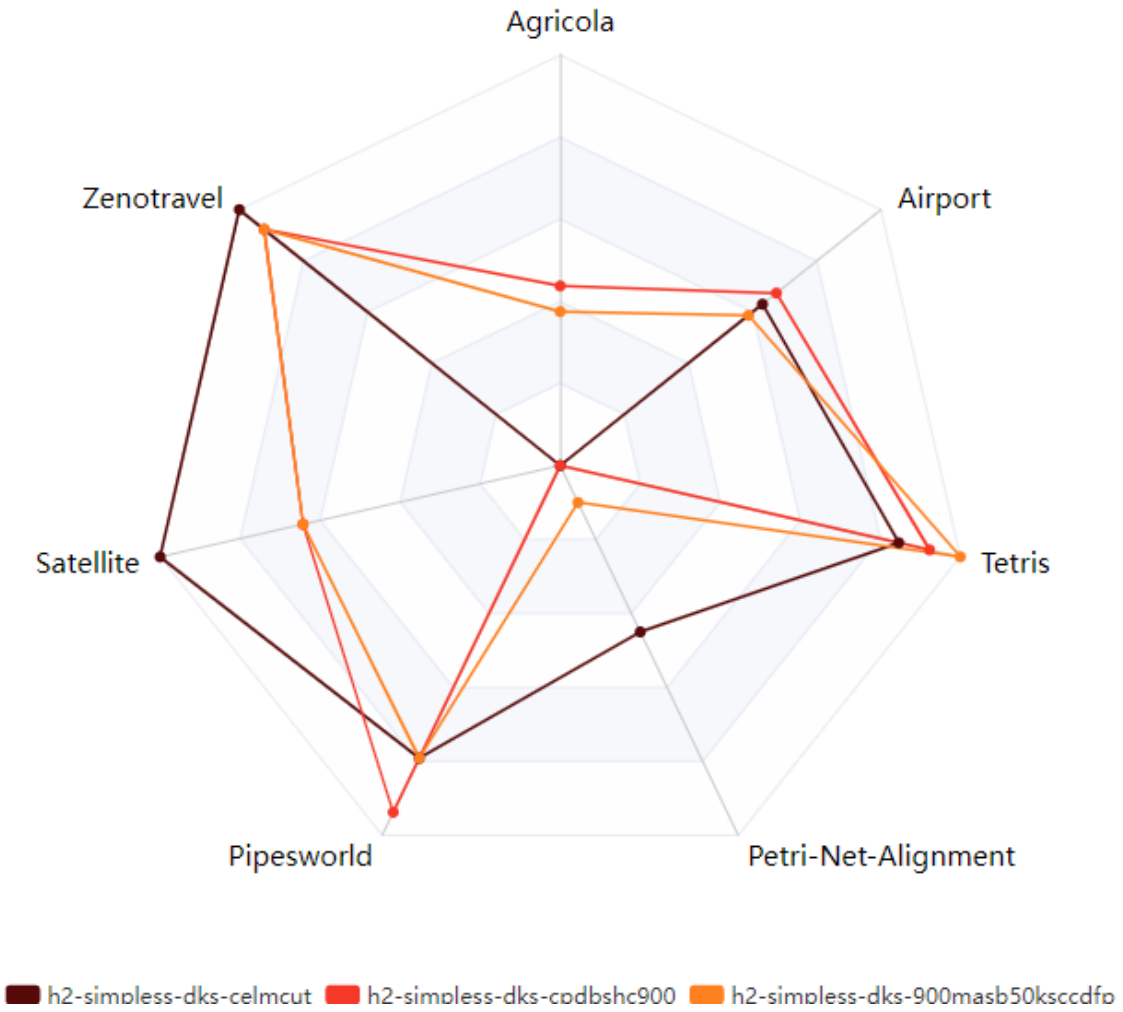
$P_{PDDL} = (O, P, A, s_1, \delta)$

- Consists of objects O, predicates P, action schemas A, an initial state s_1 and a set of goal conditions δ
- Predicates describe relationships between objects
- Action schemas potentially change the relationships described between objects
- The initial state is given by predicates depicting certain relationships between objects
- All states that adhere to the goal conditions are goal states



Initial State Goal State

Why Planner Portfolios?



Offline Portfolios and Online Portfolios

Offline Portfolios:

- An offline portfolio is a schedule of planners paired with time allocations
- Is trained to produce a sequence
- Does not take task specific information into account
- Doesnt require extra computational overhead for each task

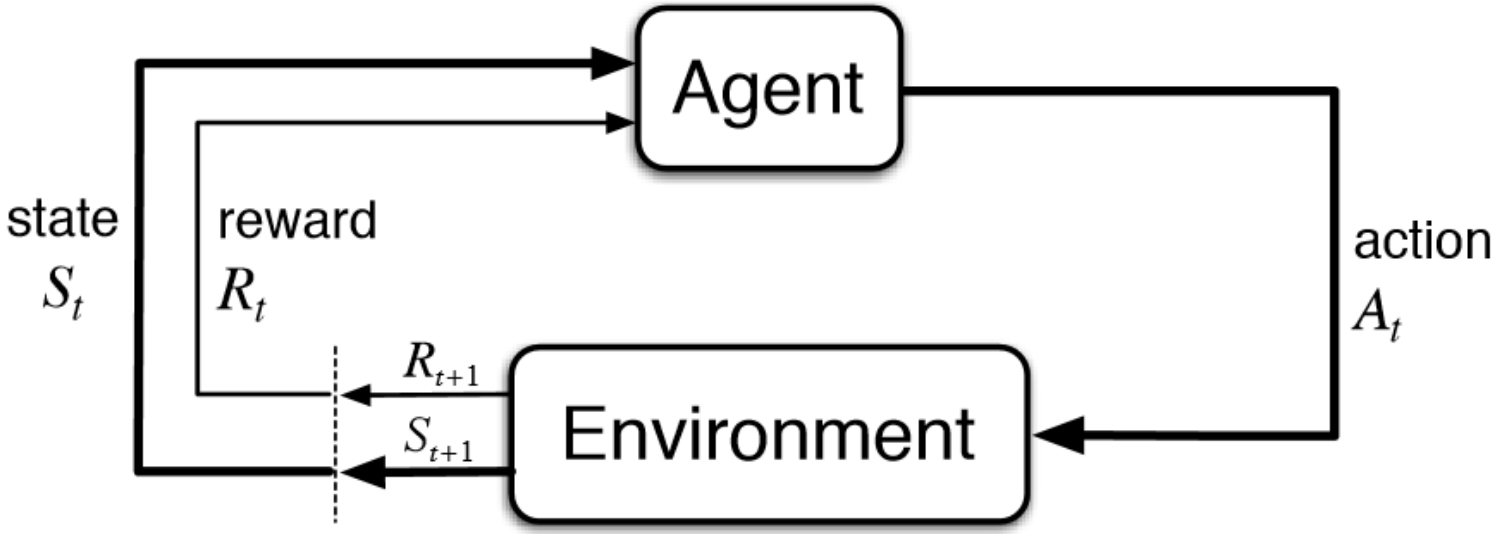
Online Portfolios:

- An online portfolio is a function with input being the current task and a history of attempted planners and the output is a planner time allocation pair
- Is trained to make predictions
- Takes task specific information into account
- Requires extra computational overhead for each task

Previous Work on Online Portfolios

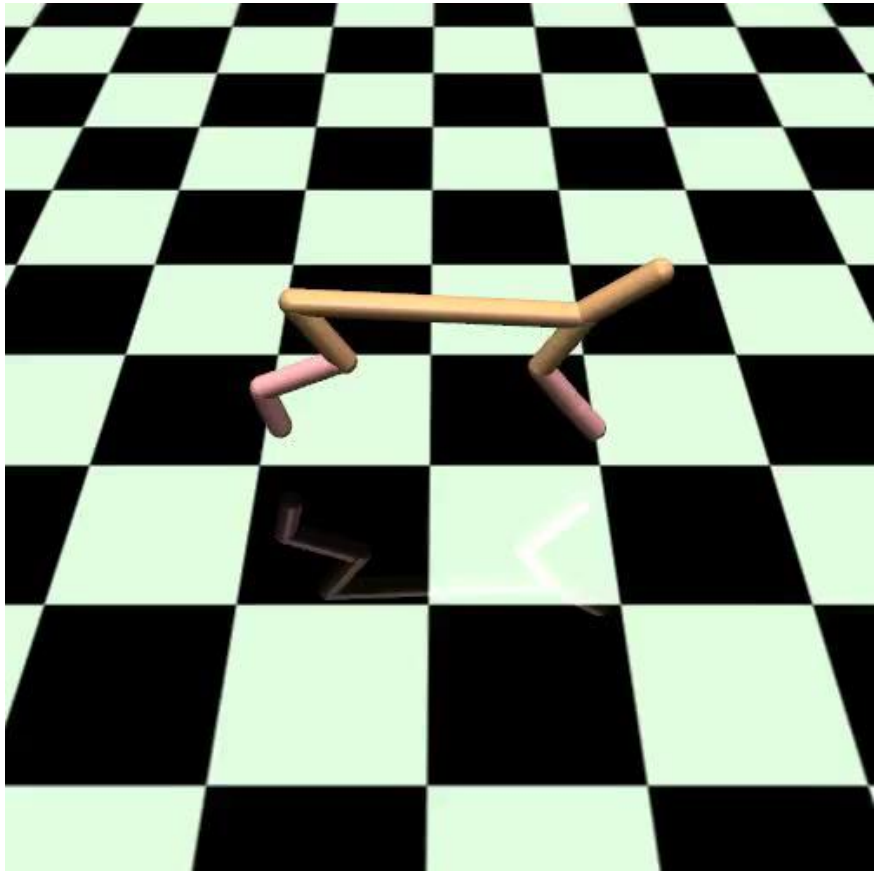
- Delfi (Katz et al., 2018; Sievers et al., 2019a)
- Used Supervised Learning
- Had to train multiple networks to make multiple predictions
- Additional networks require progressively longer to train
- Doesn’t scale well

Why use Reinforcement Learning for Online Planner Portfolios?



Reward Functions & Reward Shaping

- Reward is a metric for quality of an action given a certain state
- Reward determines how & what agent learns
- Proper reward construction integral to agent's success



Reward Functions & Reward Shaping

- Sparse rewards → slow learning
- Shaped rewards → faster learning , local optima potential
- Extreme reward shaping (dense rewards) → much faster learning, even more local optima potential
- Reward below is the reward for our reward shaping DDPG

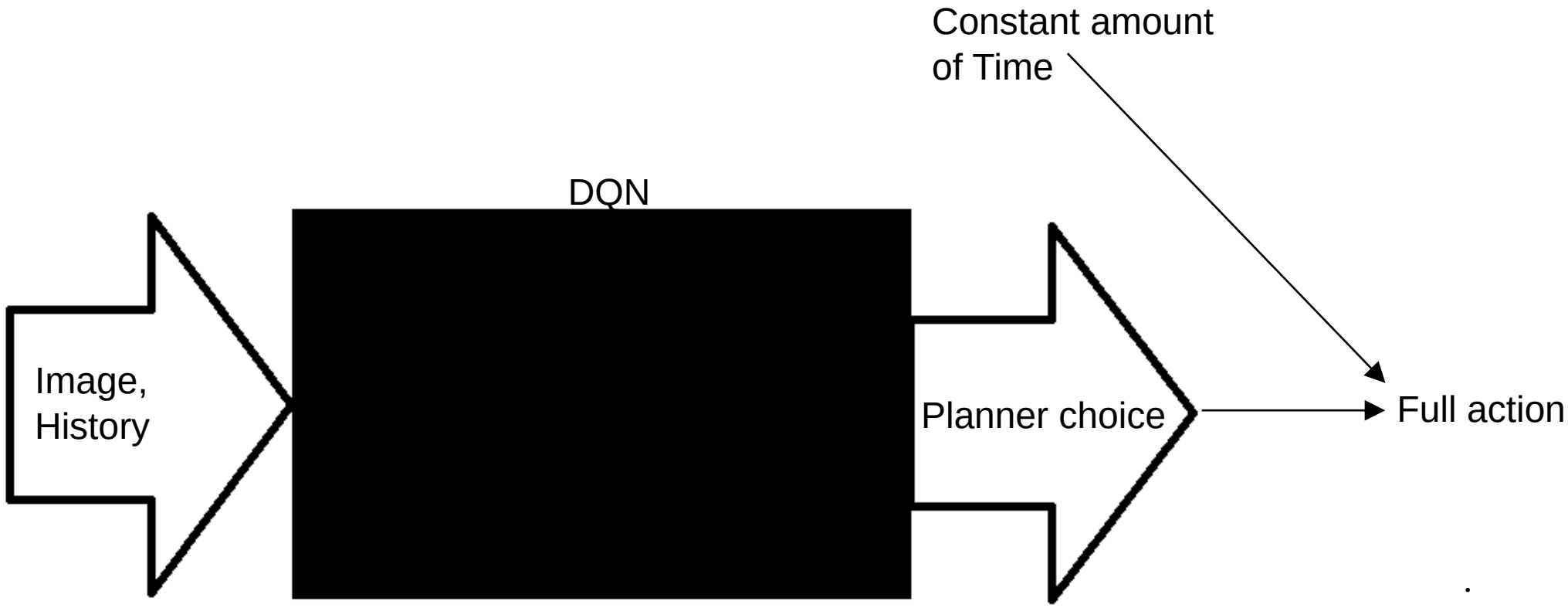
$$R = \begin{cases} -11 & \text{iff no planner can solve in timeLeft any more} \\ -10 & \text{iff plannerTime} \leq 0 \\ -1 & \text{iff some planners can still solve} \\ 1 + 10 * \frac{timeLeftEpisode}{timePerEpisode - timeForBestPlanner} & \text{iff current planner and time allocation solves task} \end{cases}$$

Bellman equation & Deep Q-Networks

$$V(s) = max\{R(s,a) + \gamma V(s')\}$$

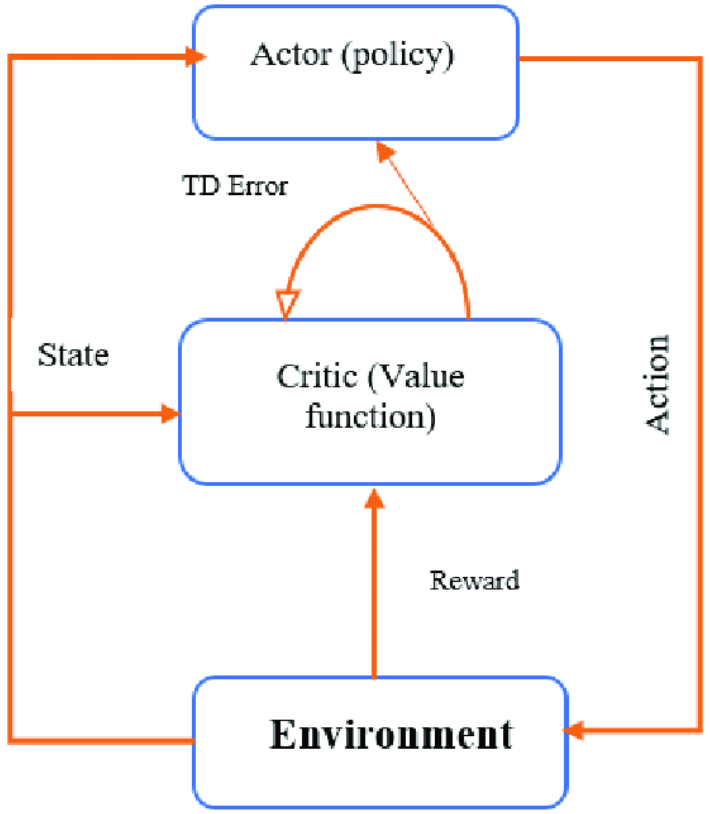
- DQN is a RL algorithm for discrete action spaces that combines Q-Learning with deep neural networks
- Uses Bellman equation to calculate expected Q-values
- Network determines Q-values of state action pairs
- Selects action with highest Q-value for given state

Our DQN

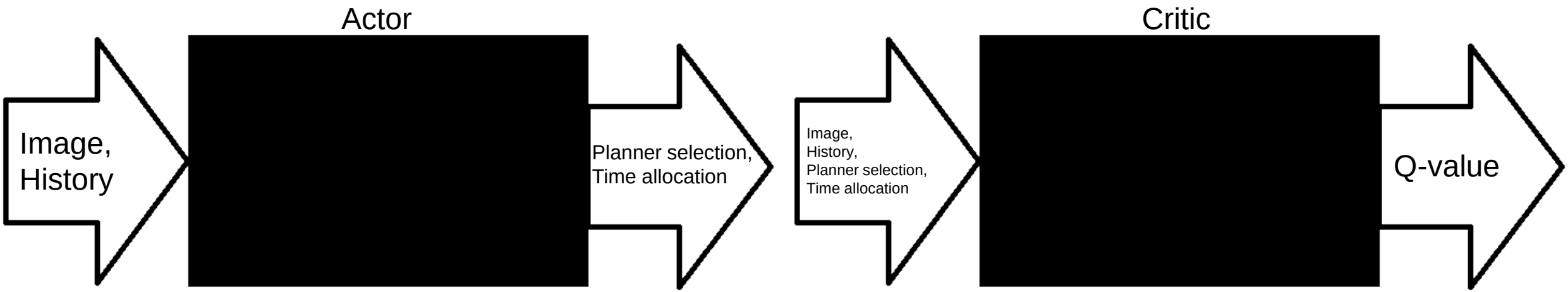

Constant amount
of Time
DQN
Image,
History
Planner choice
Full action

Actor-Critic methods & Deep Deterministics Policy Gradients

- Consist of Actor & Critic networks
- Actor network estimates «continous» action
- Critic networks estimates Q-value of state action pair
- In this project only Actor Critic used DDPG



Our DDPG

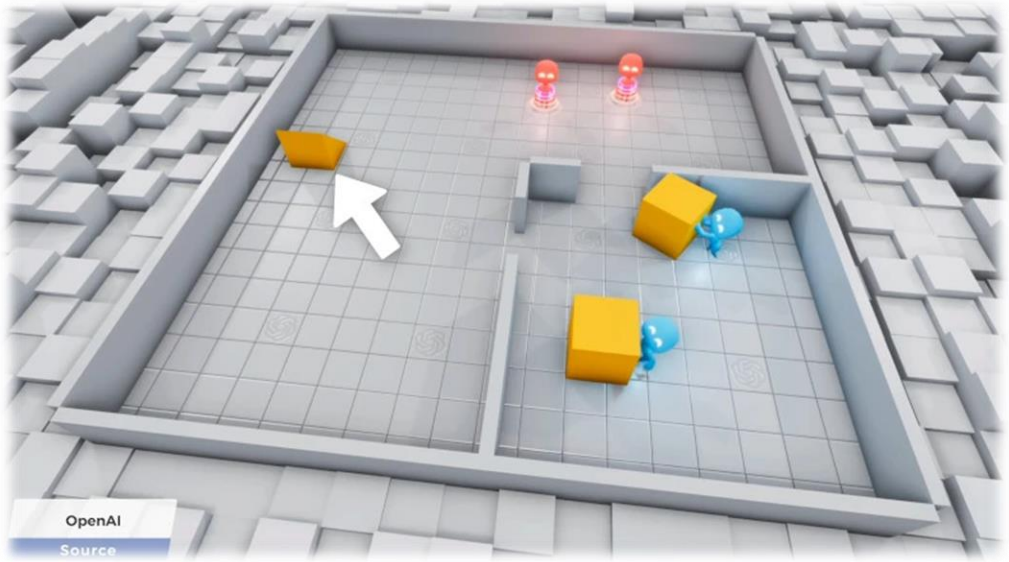

Actor
Image,
History
Planner selection,
Time allocation
Critic
Image,
History,
Planner selection,
Time allocation
Q-value

Discrete-Continous Hybrid Action Spaces

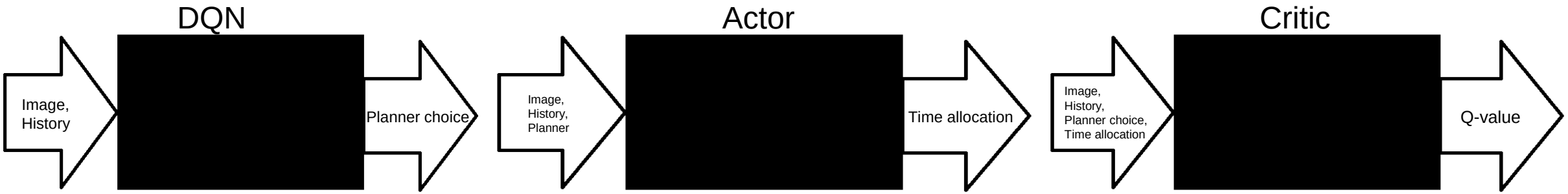
- Hybrid action spaces have discrete and continous action components
- Most RL agents made to work on either continous or discrete action spaces
- Difficult to deal with combination
- Multiple agents interacting in a single environment → multi-agent RL

Multi-Agent Reinforcement Learning

- Multiple agents interacting in the same environment
- Cooperative MARL v.s. Adversarial MARL
- Cooperative MARL → e.g., agents learn to play hide and seek together (Baker et al., 2019)
- Adversarial MARL → e.g., agents compete while playing hide and seek (Baker et al., 2019)
- Can we formulate online planner portfolio learning as a cooperative MARL environment?

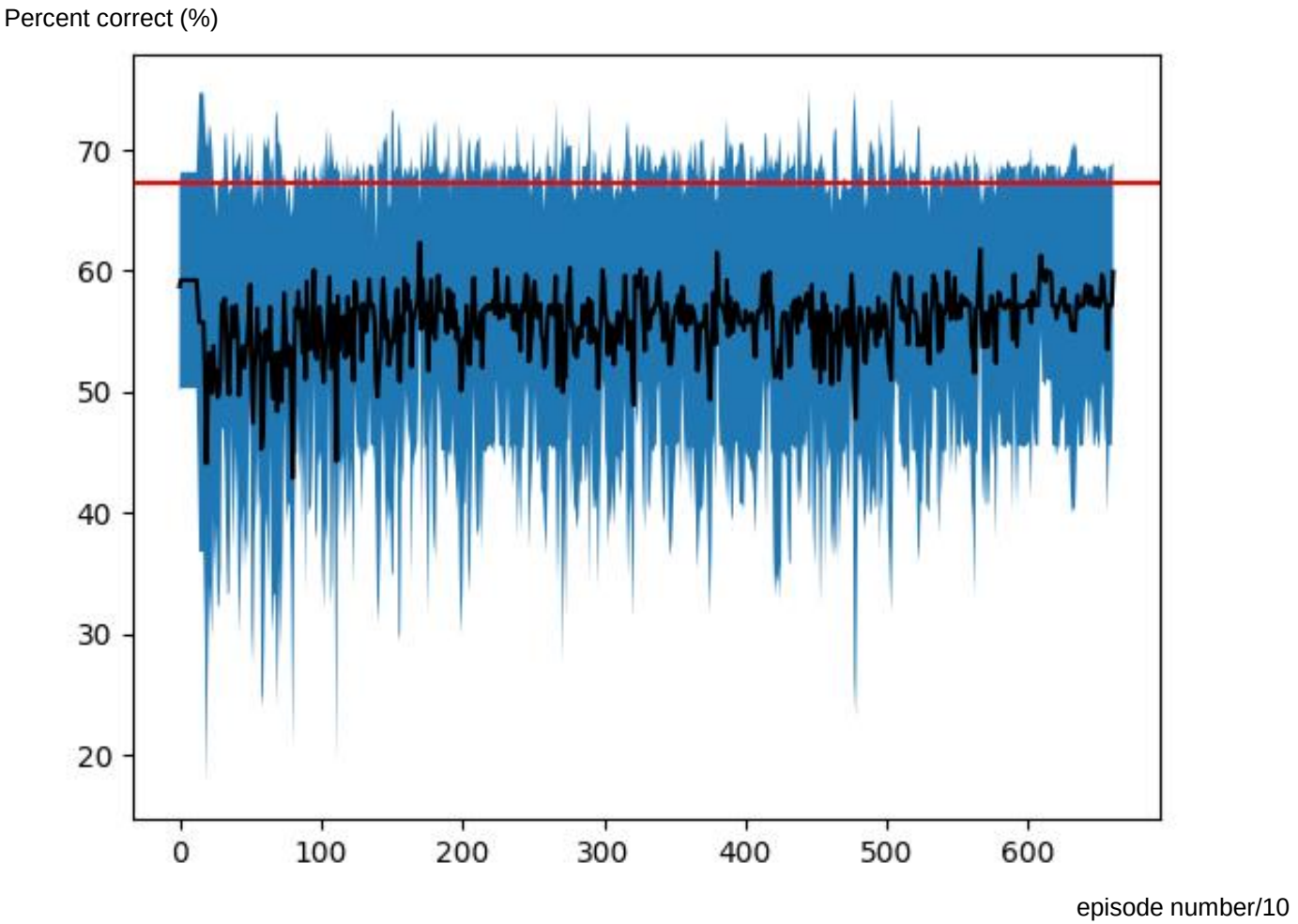


Our MARL

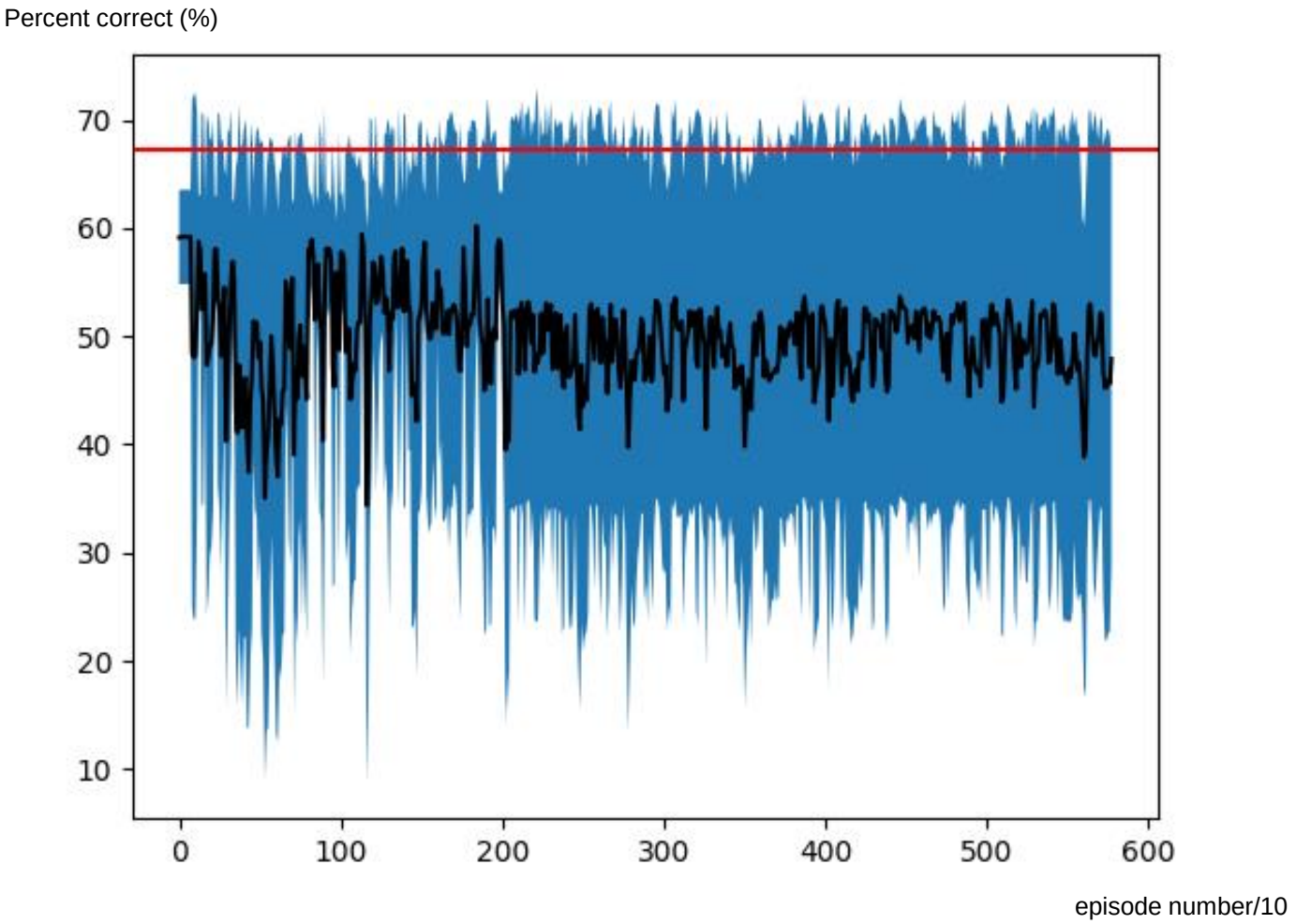

DQN
Image,
History
Planner choice
Actor
Image,
History,
Planner
Time allocation
Critic
Image,
History,
Planner choice,
Time allocation
Q-value

Experiments

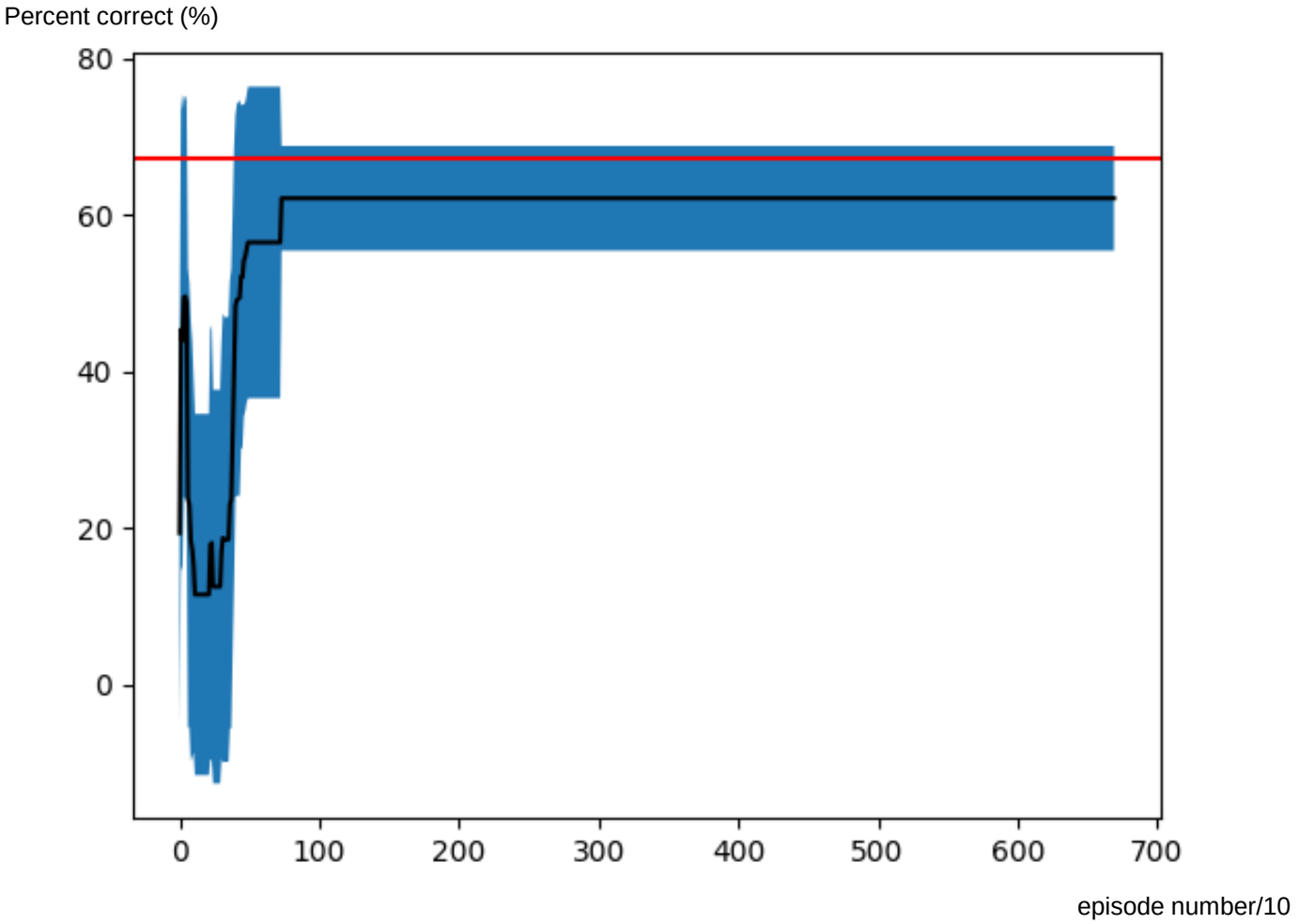
Sparse Rewards DDPG



Reward Shaping DDPG



MARL with MSE reward



Future Work

- Insufficient amount of task data → generate more task data
- Insufficient quality of samples → e.g., Prioritized Experience Replay
- Insufficient stability and sample efficiency in current RL → Supervised Learning
- Task representation optimal as input? → Research into alternative methods for encoding PDDL tasks

Conclusion

- RL approach could not outperform SL approaches such Delfi (Katz et al., 2018; Sievers et al., 2019a)
- Likely due to size and quality of tasks in data set
- For Delfi max number of predictions n=2
- n>10 likely not very helpful → maybe SL approach doesnt need to scale much

Questions?