Lecture-10
Model Based Learning

Outline

- Model-based RL
- Dyna
- Dyna Plus

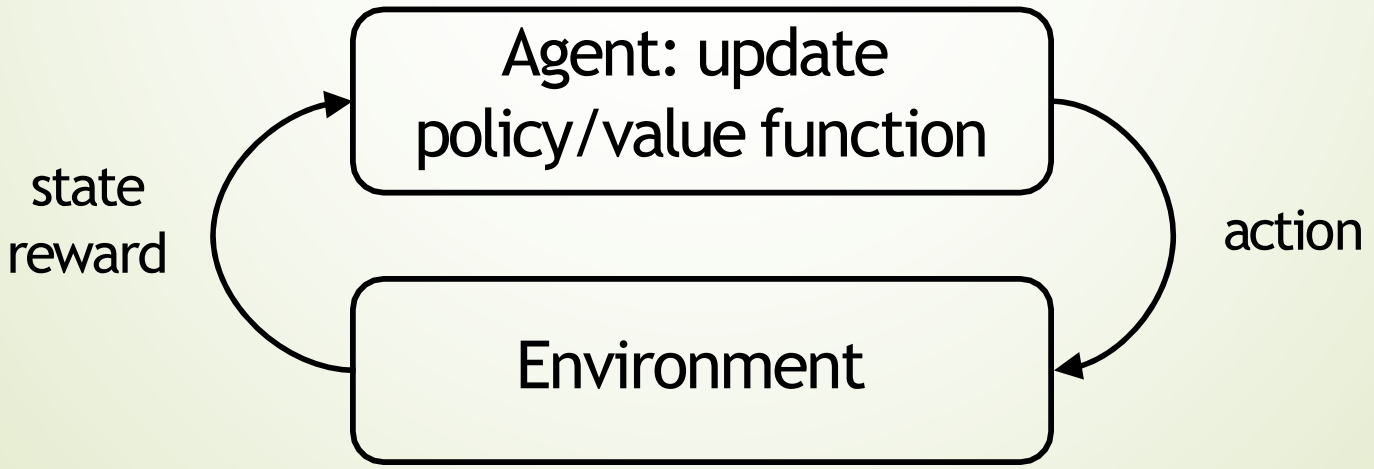
Introduction…

- By a *model* of the environment we mean anything that an agent can use to predict how the
  - environment will respond to its actions.
- Given a state and an action, a model produces a
  - prediction of the resultant next state and next reward.
- If the model is stochastic, then
  - there are several possible next states and next rewards, each with some probability of occurring.
- Some models produce a description of all possibilities and their probabilities these we call *distribution models*.

- we say the model is used to *simulate* the environment and produce *simulated experience*.

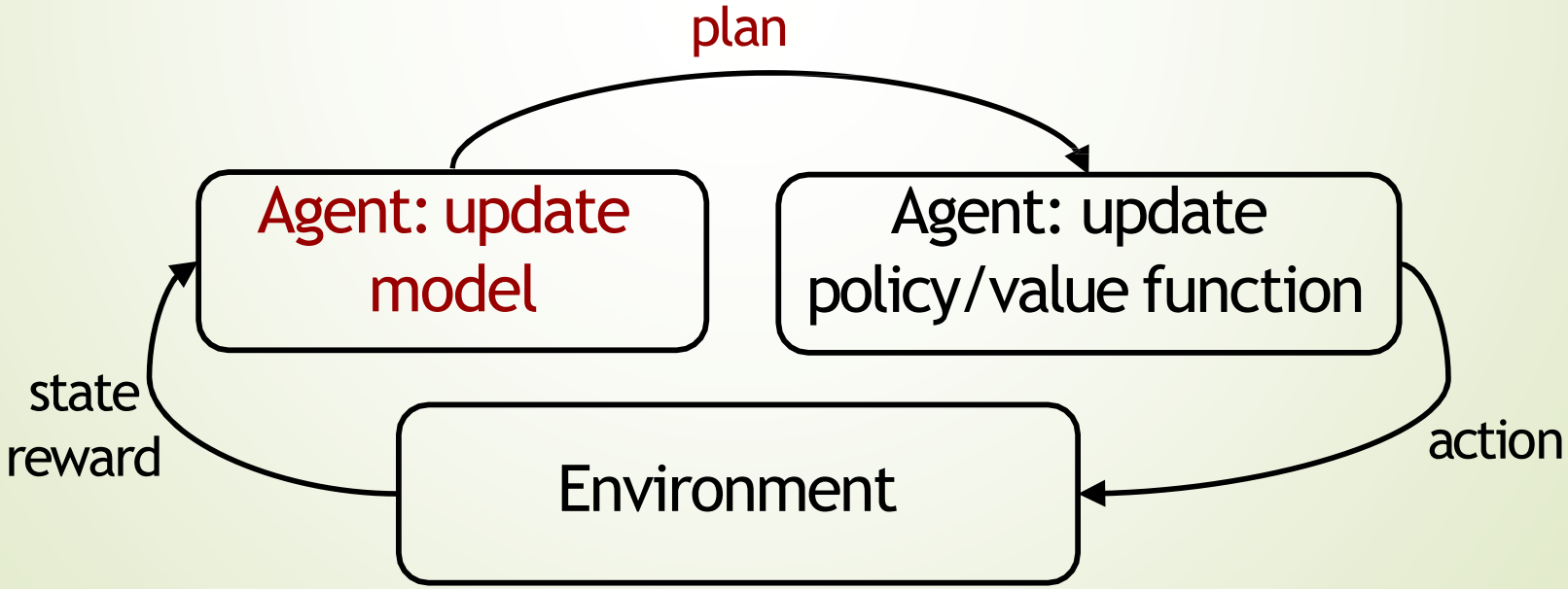
Model-free RL

- No explicit transition or reward models
  - Q-learning: **value-based method**
  - Policy gradient: **policy-based method**
  - Actor critic: **policy and value based method**



Model-based RL

- **Learn explicit transition and/or reward model**
  - **Plan based on the model**
  - Benefit: Increased sample efficiency
  - Drawback: Increased complexity

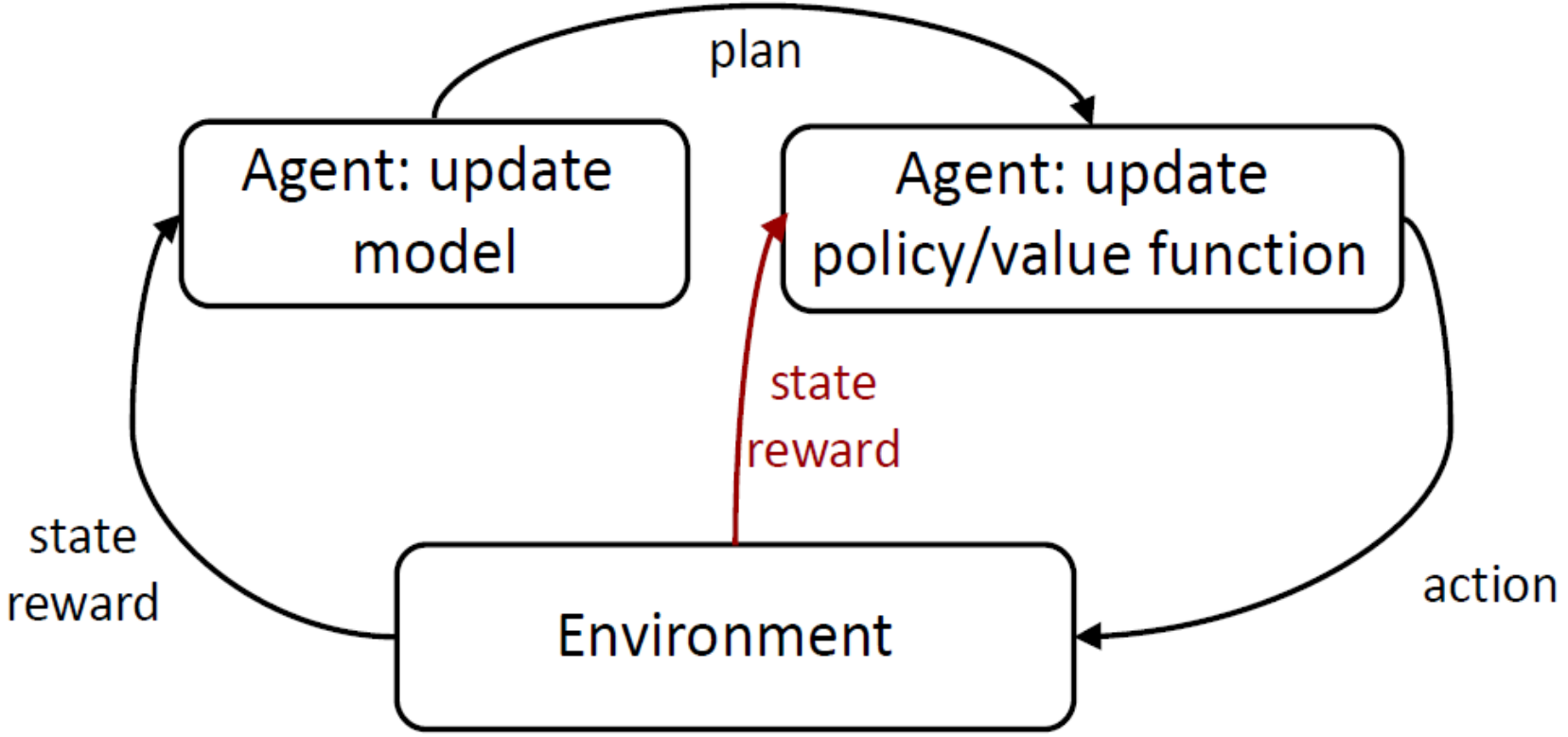


Model-based RL

- Idea: at each step
  - Execute action
  - Observe resulting state and reward
  - Update transition and/or reward model
  - Update policy and/or value function

Dyna

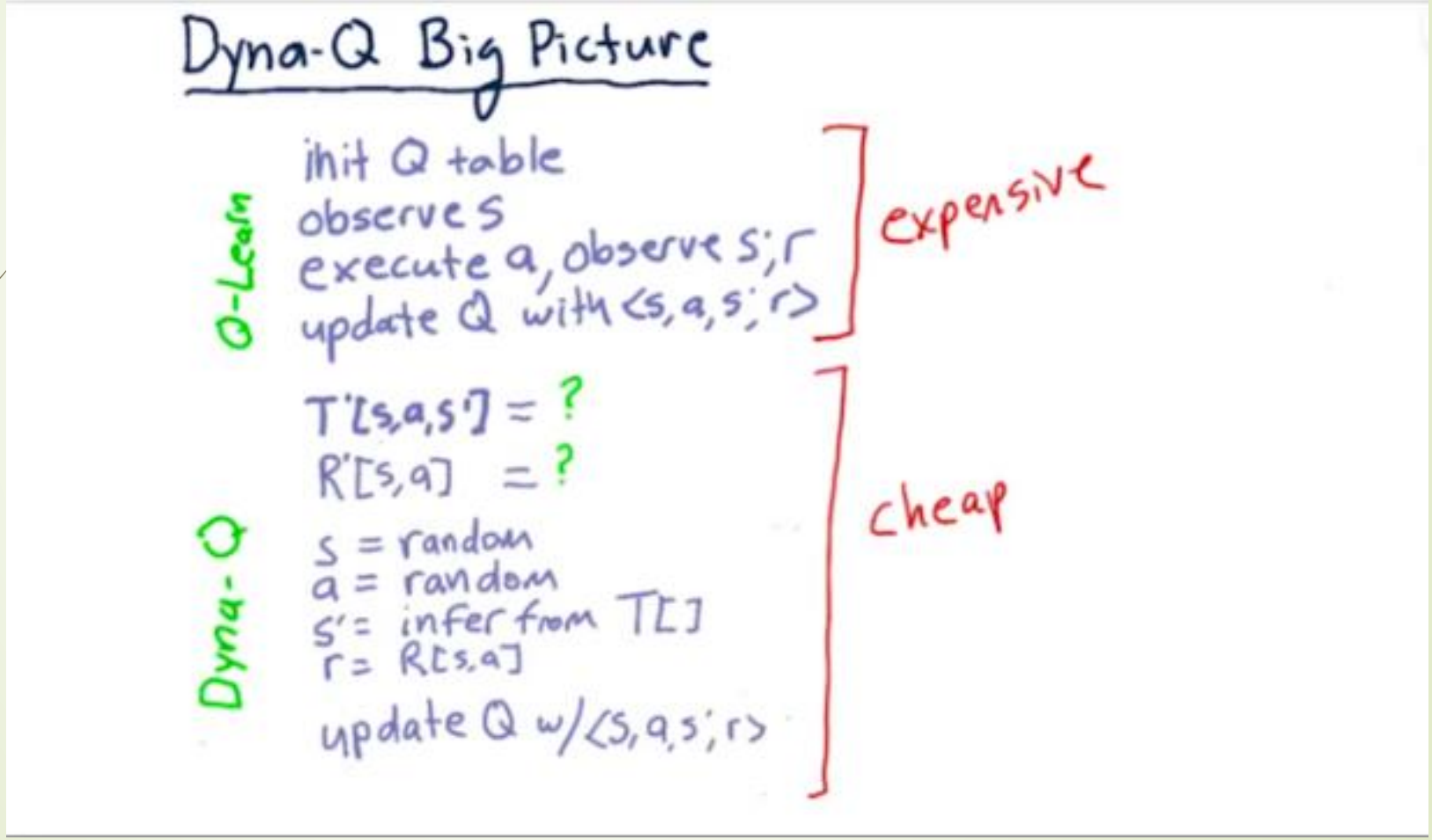
- **Learn explicit transition and/or reward model**
  - Plan based on the model
- **Learn directly from real experience**



Why we choose Dyna?

- One problem with Q-learning is that it takes many experience tuples to converge. Gathering experience from the real world is expensive because it requires us to take real action - executing a live trade in our case - to gather information.
- To address this problem, researchers developed a technique called Dyna. Dyna works by building models of the transition function, *T*, and reward function, *R*. Then, after each real interaction with the world, we hallucinate many additional interactions, typically a few hundred, which we use to update the Q-table.

Dyna-Q Big Picture

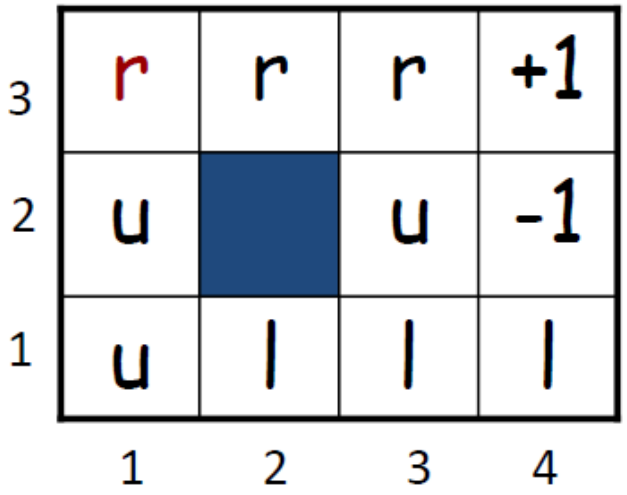


https://www.omscs-notes.com/machine-learning-trading/dyna/

Pseudo code of Tabular Dyna-Q is shown as follow.

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow \varepsilon$-greedy(S,Q)
- (c) Take action A; observe resultant reward, R, and state, S'
- (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
- (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
- (f) Loop repeat n times:
  - $S \leftarrow$ random previously observed state
  - $A \leftarrow$ random action previously taken in S
  - $R, S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

Maze Example



$\gamma = 1$

Reward is -0.04 for non-terminal states

We need to learn all the transition probabilities!

(1,1)→ (1,2)→ **(1,3)**→ (1,2)→ **(1,3)**→ (2,3)→ (3,3)→ $(4,3)_{+1}$
(1,1)→ (1,2)→ **(1,3)**→ (2,3)→ (3,3)→ (3,2)→ (3,3)→ $(4,3)_{+1}$
(1,1)→ (2,1)→ (3,1)→ (3,2)→ $(4,2)_{-1}$

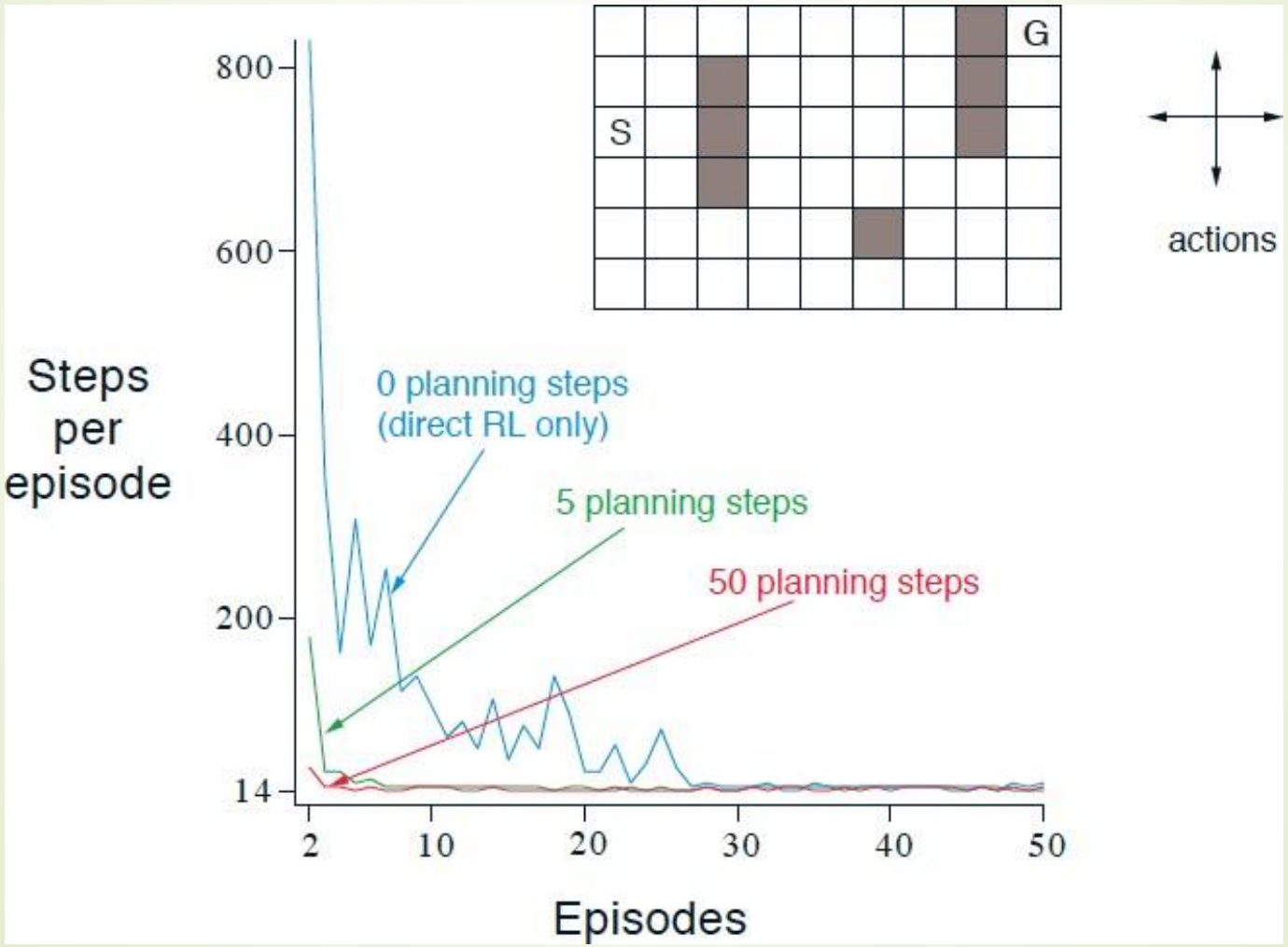
P((2,3)|(1,3),r) =2/3
P((1,2)|(1,3),r) =1/3

} Use this information in

$$V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} \Pr(s'|s,a)\, V^*(s')$$

Dyna-Q

- Task: reach G from S

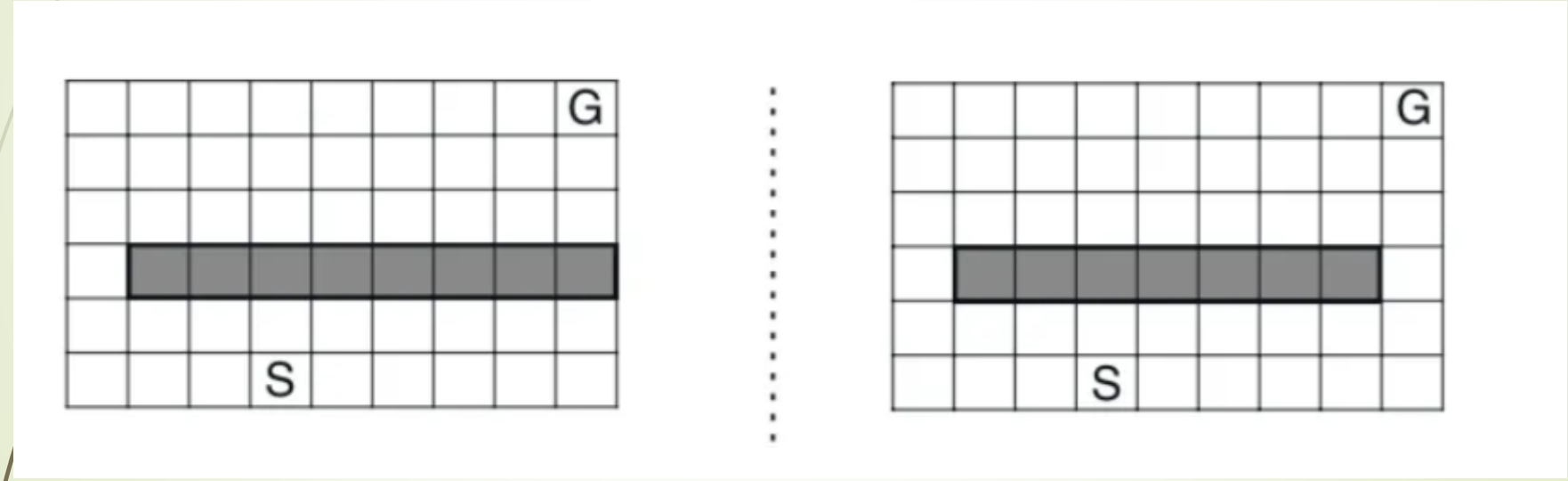


Two Problems with DYNA-Q

- What if the model is wrong?
- How to update the Q function more efficiently?

Models may be incorrect because the environment is stochastic and only a limited number of samples have been observed, or because the model was learned using function approximation that has generalised imperfectly, or simply because the environment has changed and its new behaviour has not yet been observed. When the model is incorrect, the planning process is likely to compute a suboptimal policy.

Shortcut Maze



Solution: DYNA-Q+

The agent keeps track for each state–action pair of how many time steps have elapsed since the pair was last tried *in a real interaction with the environment. The more time that has elapsed, the greater (we might presume) the chance that the dynamics of this pair has changed and that the model of it is incorrect.* ***To encourage behaviour that tests long-untried actions, a special "bonus reward" is given on simulated experiences involving these actions.*** *In particular, if the modelled reward for a transition is* r*, and the transition has not been tried in* τ *time steps, then planning updates are done as if that transition produced a reward of* $r + \kappa\sqrt{\tau}$*, for some small* κ*. This encourages the agent to keep testing all accessible state transitions and even to find long sequences of actions in order to carry out such tests.*

Summary:

- To summarise, the algorithm is totally the same as Dyna-Q except that **it keeps track of number of time a state has been visited, and give reward to state that has long not been visited**(as these state could have possibly changed as time goes on).