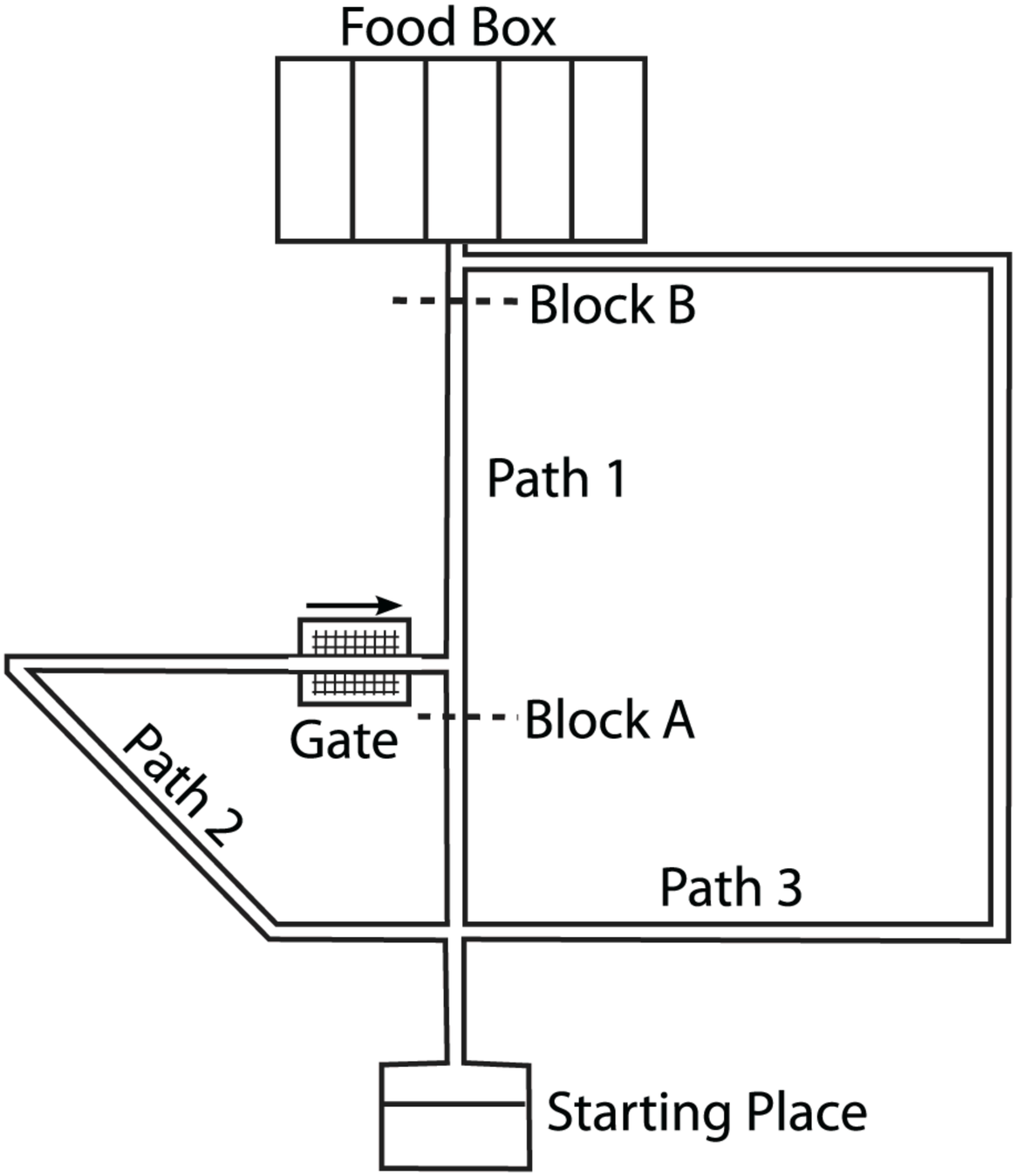
Learning, Planning, and Acting

CMPUT 655
Fall 2022

Rats learn models



How do I submit my project proposal?

Sign up on the gsheet. That’s it!

Planning in action

https://github.com/AmiiThinks/rltoolkit

What if there's a trip-wire square that opens/closes a shorter path ….

Dyna assumes a Markov state! If you don’t have that you need to combine Dyna with representation learning!

Sample Models

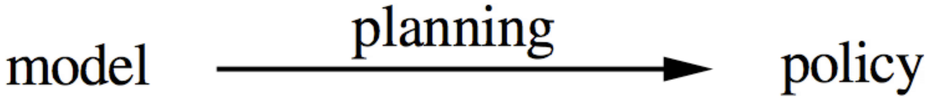
- Model: anything the agent can use to predict how the environment will respond to its actions
- Sample model, a.k.a. a simulation model
  - produces sample experiences for given s, a
    - sampled according to the probabilities
  - allows reset, exploring starts
  - often much easier to come by
- Both types of models can be used mimic or simulate experience: to produce hypothetical experience

Models

- Consider modeling the sum of two dice
  - A *distribution model* would produce all possible sums and their probabilities of occurring
  - A *sample model* would produce an individual sum drawn according to the correct probability distribution
- When we solved the Gambler's problem with value iteration, we used the distribution model
- When you solved the Gambler's problem with Monte-Carlo, you implemented a sample model in your environment code

Planning

- Planning: any computational process that uses a model to create or improve a policy



- We take the following (unusual) view:
  - update value functions using both real and simulated experience
  - all state-space planning methods involve computing value functions, either explicitly or implicitly
  - they all apply updates from simulated experience

One-step Tabular Q-planning

Random-sample one-step tabular Q-planning

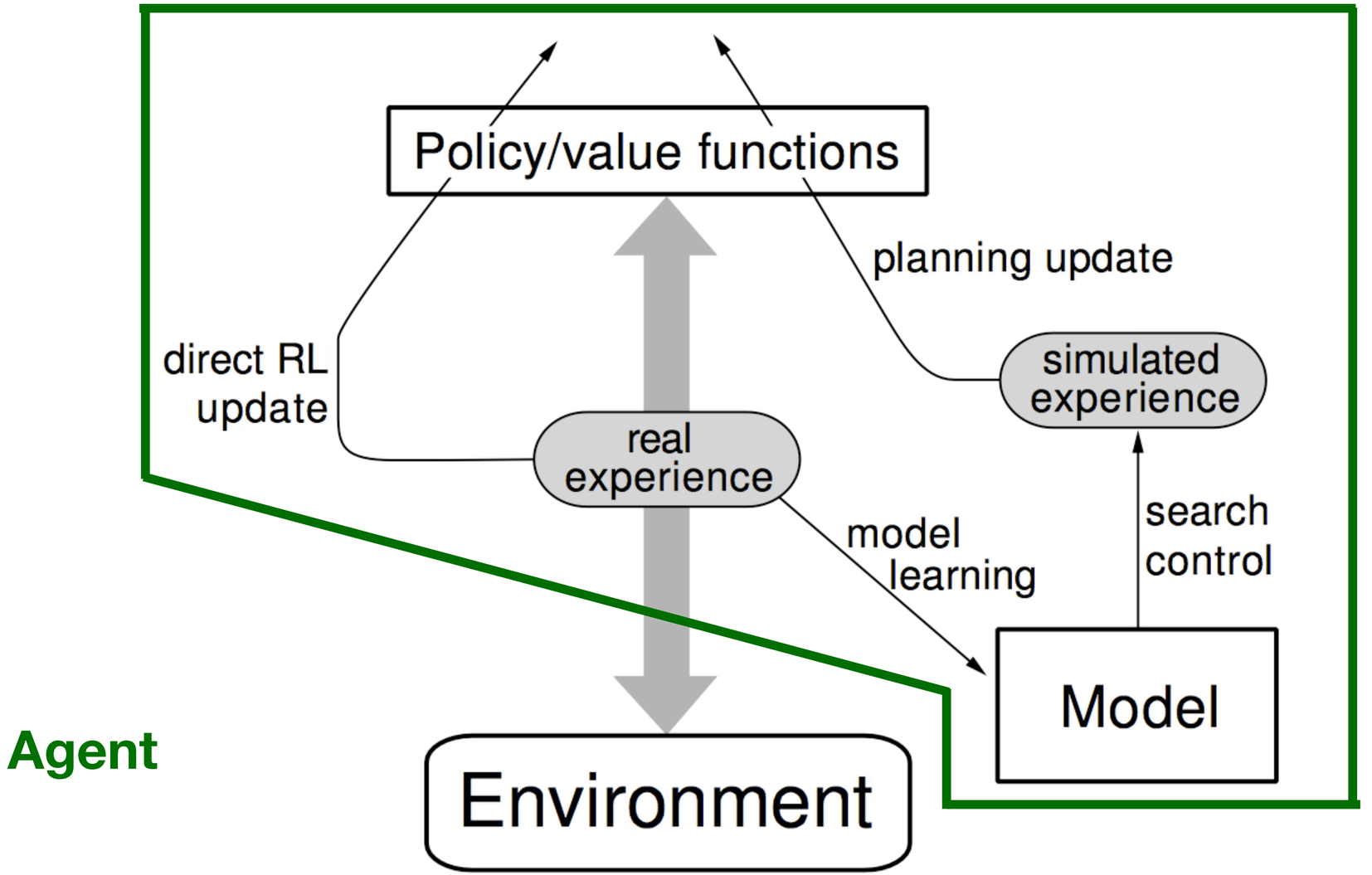
Loop forever:

1. Select a state, $S \in \mathcal{S}$, and an action, $A \in \mathcal{A}(S)$, at random
2. Send S, A to a sample model, and obtain
   a sample next reward, R, and a sample next state, S'
3. Apply one-step tabular Q-learning to S, A, R, S':
   $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$

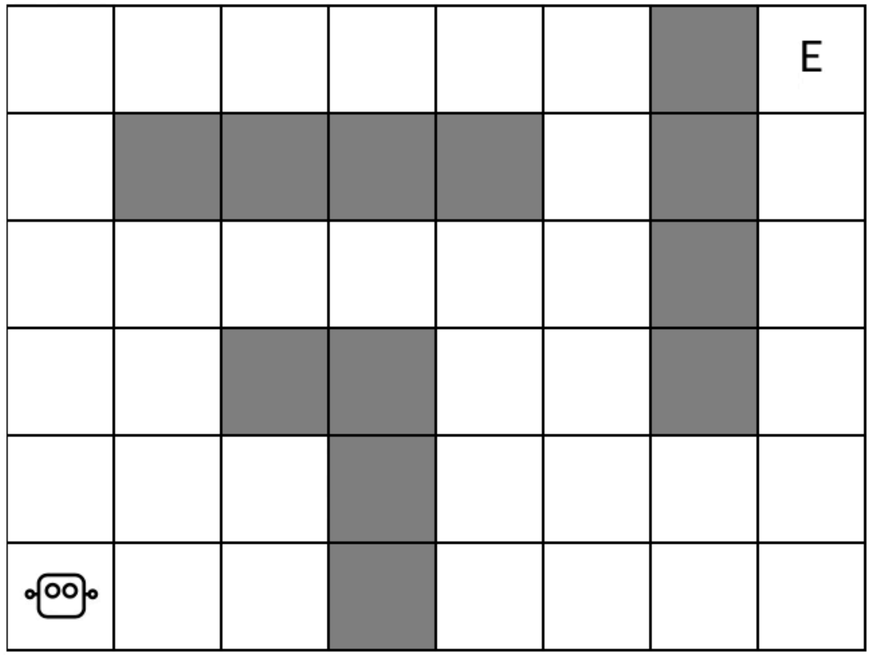
Question: How does this differ from DP, specifically Value Iteration?

Tabular Dyna-Q

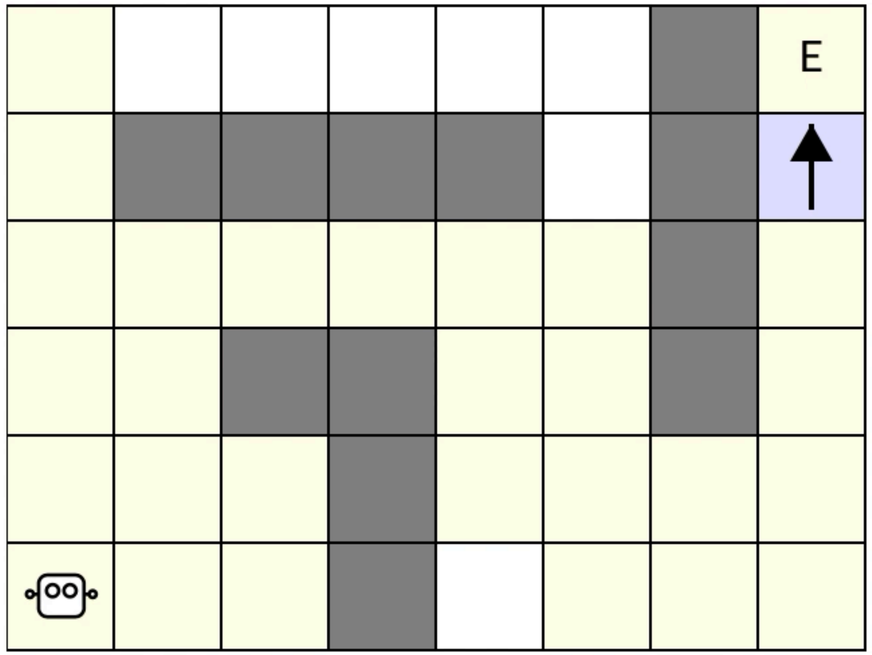
The Dyna Architecture



Number of actions taken: 0

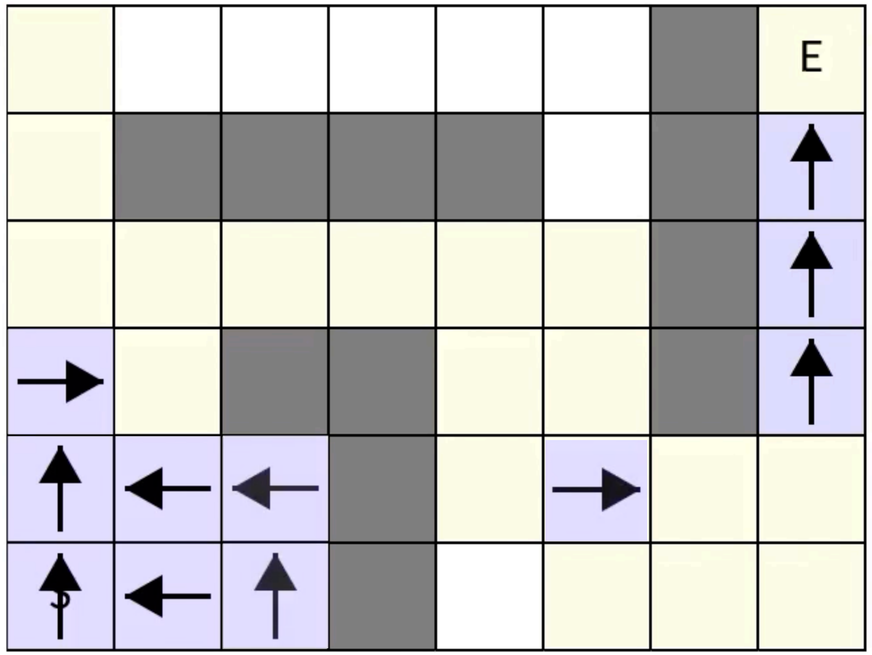


Number of actions taken: 184

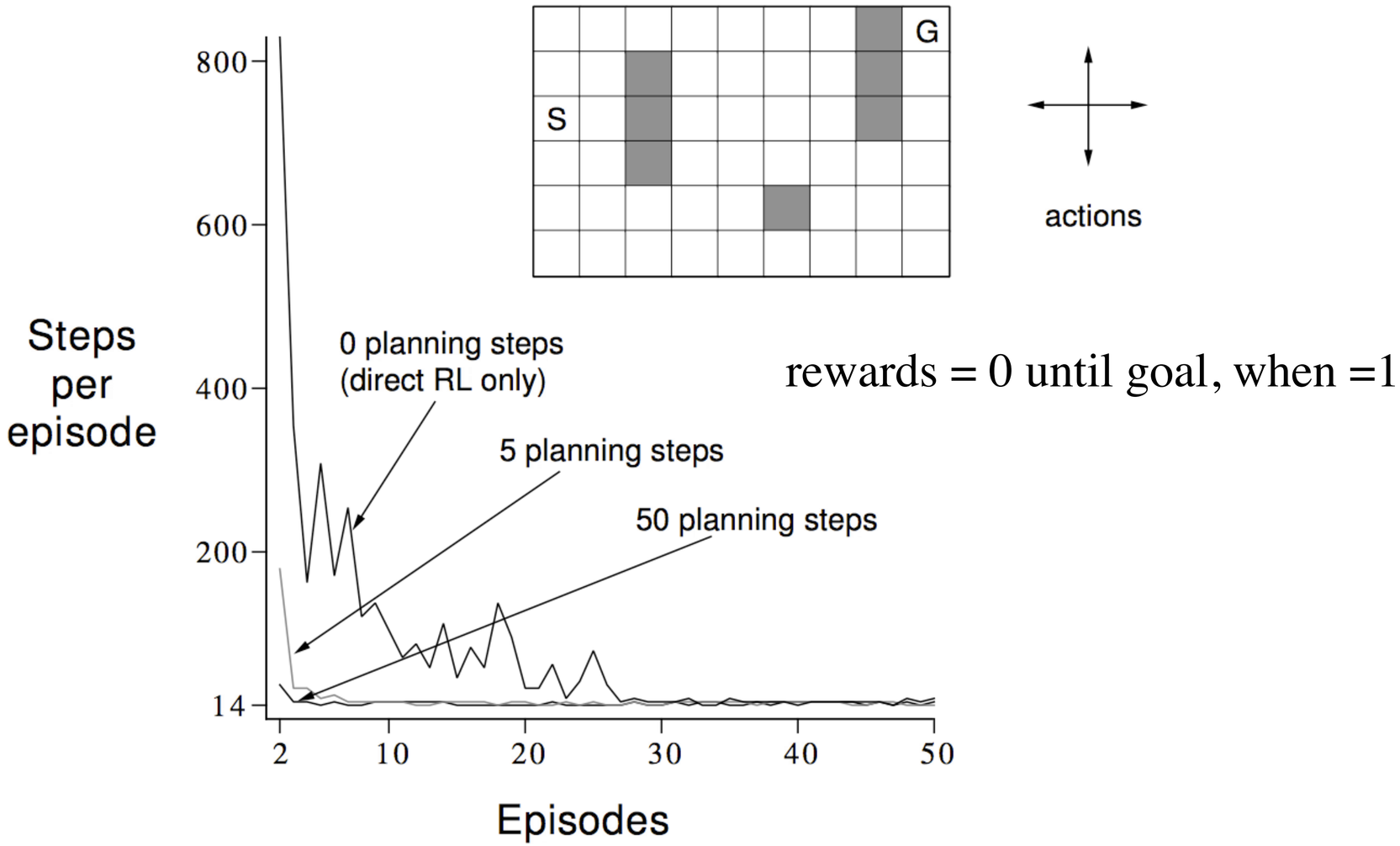


Number of steps planned: 100

Number of actions taken: 185

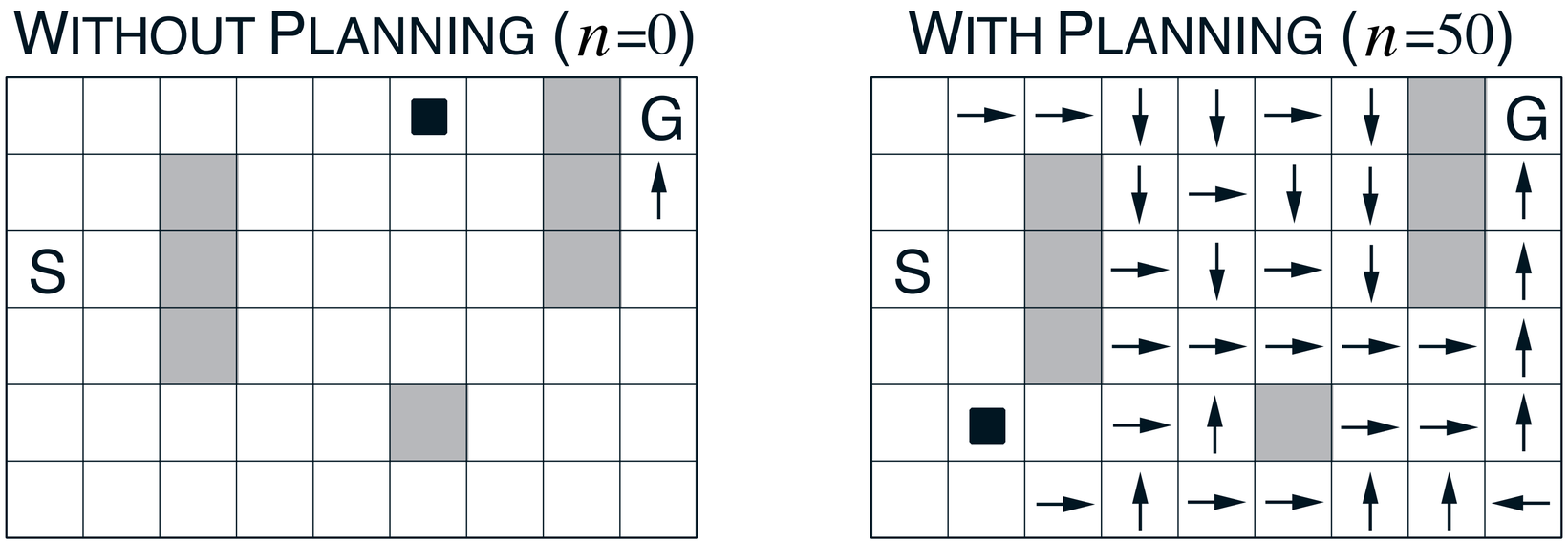


Dyna-Q on a Simple Maze



rewards = 0 until goal, when =1

Dyna-Q Snapshots: Midway in 2nd Episode



Implementation details

- Notice during the demo, that the one-step Q-learning agent and Dyna-Q agent appeared to be equally reactive
- Updating the value function and selecting a new action is very fast
  - Thus, there is usually some time left over
  - We can use that time to run a planning loop
    - Planning with n=5 helps a lot
- What are other ways we could integrate planning with learning and acting?
- Planning of this form is *anytime*

Discussion posts

- Do people ever learn and plan with distribution models
  - I believe this is done in this chapter
- What about compounding errors and rollouts
  - Big issue, see lots of good refs in here: https://arxiv.org/abs/2006.04363
  - Option models, avoiding learning transitions all together (https://arxiv.org/abs/2206.02902)
- The UCB method from chapter 2 is both conceptually and mathematically similar to the virtual bonus reward of Dyna-Q+
  - What are the differences?

- Is Dyna experience replay?
  - Yes, sort of. Depends if we are talking about state or observations
- Typical planning methods don't account for the types of symmetries in the world. For example, the value of being in a cosy spot on the couch is high regardless of where the couch is inside the room.
  - Representation learning…. Observations vs Agent State
- How do we handle stochastic problems
  - Note about mu-zero+Atari;
  - "For stochastic problems, prioritized sweeping is always done using one of the expected updates."

- Can we utilize knowledge of the goal state to improve exploration
  - Sure! But… "This example suggests that search might be usefully focused by working backward from goal states. Of course, we do not really want to use any methods specific to the idea of “goal state.” We want methods that work for general reward functions.”
- What about runtime concerns?
  - Planninng is anytime. Dyna-Q with no planning is Q-learning! Win, Win?
- When performing Q learning update rules to simulated experience, we obtain Q planning. However, why is it that when planning is done with real experience, like in Dyna Q, it is called ‘online planning’ as opposed to ‘model based learning’?
  - Only simulation (imagining) vs simulation and interaction

- Can we include uncertainty estimates in the model
  - Yes! https://www.sciencedirect.com/science/article/pii/S0022000008000767
- There are many ways to think about how to combine models, planning, learning and acting right?
  - Model predictive control; Decision time planning (Ala MCTS), background planning like Dyna, Experience replay, Backward models, Goal-space planning etc etc
- Thinking about a gridworld I can come up with a better method that exploits knowledge of goals, that we have a grid, etc
  - What if I told you Dyna was more general, and that's what we want
  - Easy to confuse Deterministic Dyna! kept simple so we can explore a simplified problem settings well.

- Why does the bonus in Dyna-Q+ use a sqrt
  - Related to variance
- In Dyna-Q+ algorithm, can we use some kind of switch to balance planning and real-world experience in non-stationary environments? So that our agent spends more time learning about the environment to be able to plan more efficiently.
  - Who decides the interaction rate of the problem?
  - If there is extra time, we could do more planning or just wait and do nothing?
- Can we make Dyna more like decision time planning?
  - Yep, on policy trajectory sampling. In the book
  - However, the anytime aspect of Dyna is very nice

- How do we set kappa in Dyna-Q+
  - Grid search? Other ideas?
- What if exploring is dangerous, how do we adapt Dyna-Q+ to handle this
  - The idea of RL is we just learn values of states. Episodes are not death. Bad things should have huge negative rewards
- Can we instead construct an MDP from experience by computing the reward distribution and transition probabilities and then using DP for solving that? Wouldn't it be more efficient?
  - Could we do this incrementally; would the agent have to pause for a long time?

What is Dyna-Q+?

- Uses an "exploration bonus":
  - Keeps track of time since each state-action pair was tried for real
  - An extra reward is added for transitions caused by state-action pairs related to how long ago they were tried: the longer unvisited, the more reward for visiting

$$R + \kappa\sqrt{\tau}$$

time since last visiting the state-action pair

  - The agent actually "plans" how to visit long unvisited states

Why does Dyna Q+ encourage exploration?

What if instead we just used the bonus in action selection?

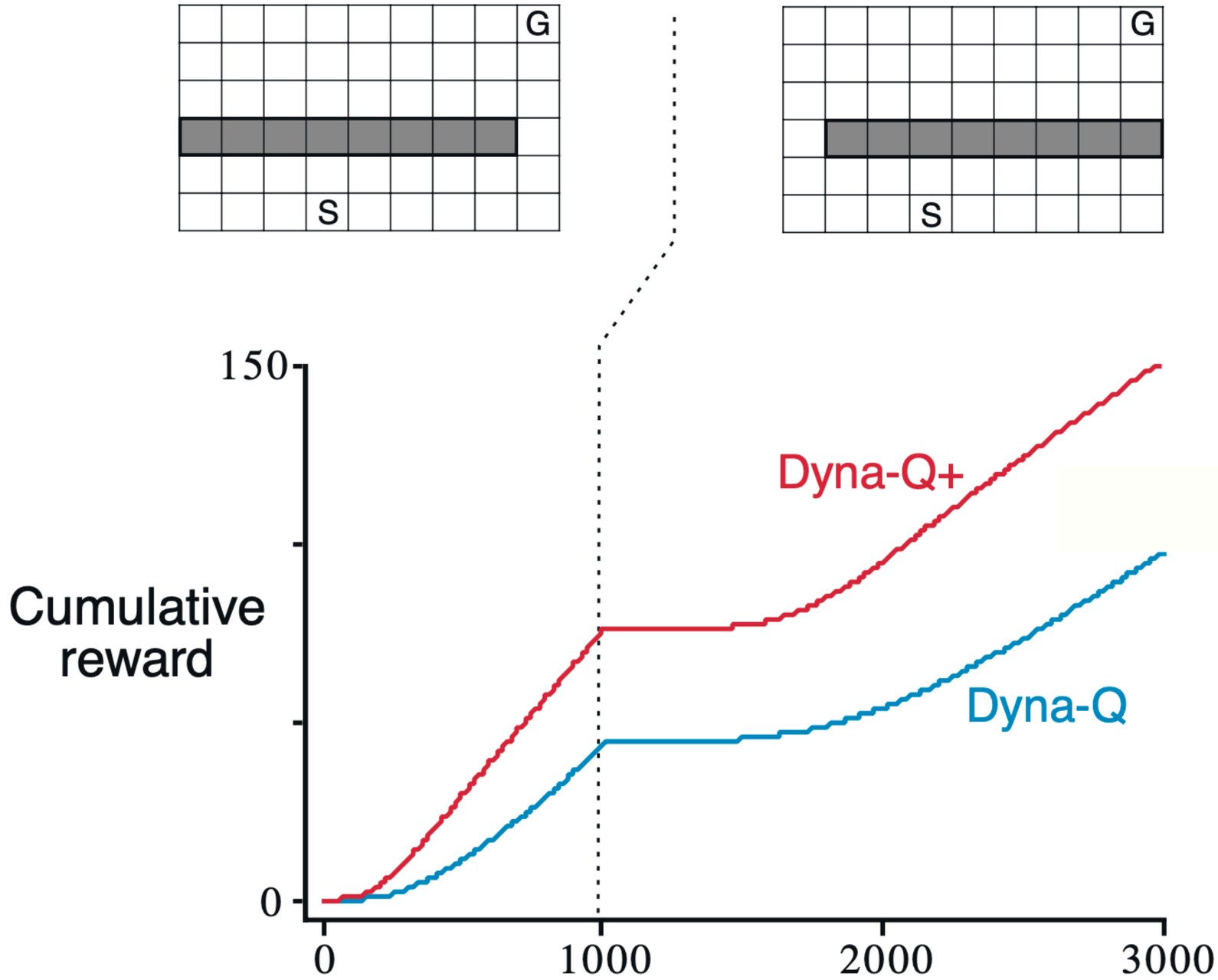
$$Q(S_t, a) + \kappa\sqrt{\tau(S_t, a)}$$

Greedy wrt to this value+bonus in step (b)

Tabular Dyna-Q

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) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
- (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
  - $R, S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Changing environments



The conflict between exploration and exploitation

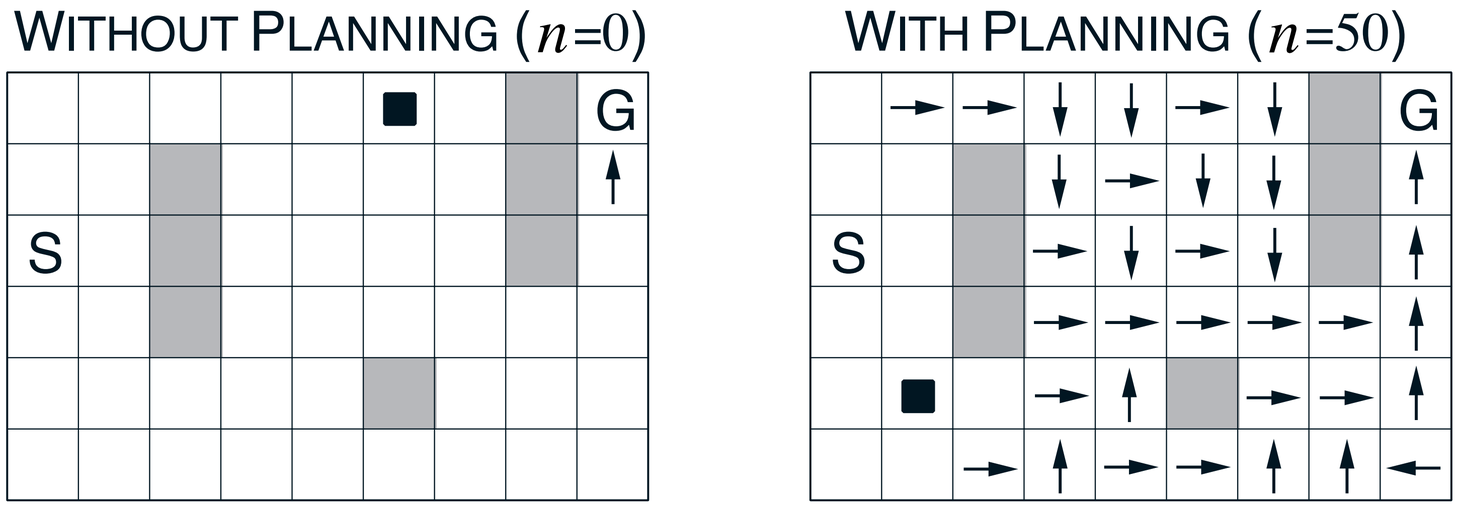
- Exploration in planning: trying actions that improve the model
  - Make it more accurate
  - Make it a better match with the environment
  - Proactively discover when the model is wrong

- Exploitation: behaving optimally with respect to the current model

- Simple heuristics can be effective

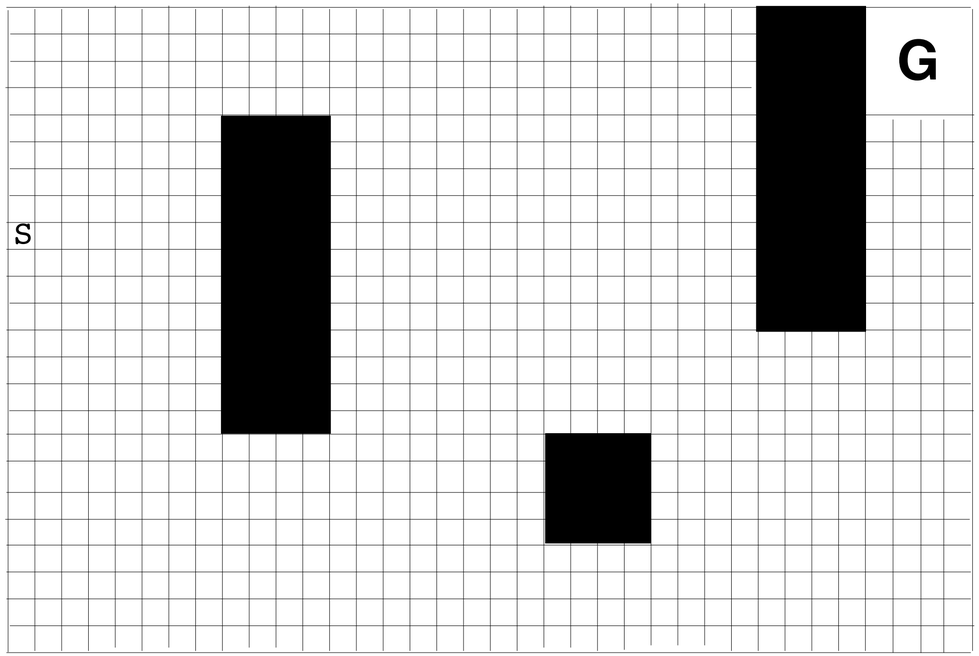
Prioritizing Search Control

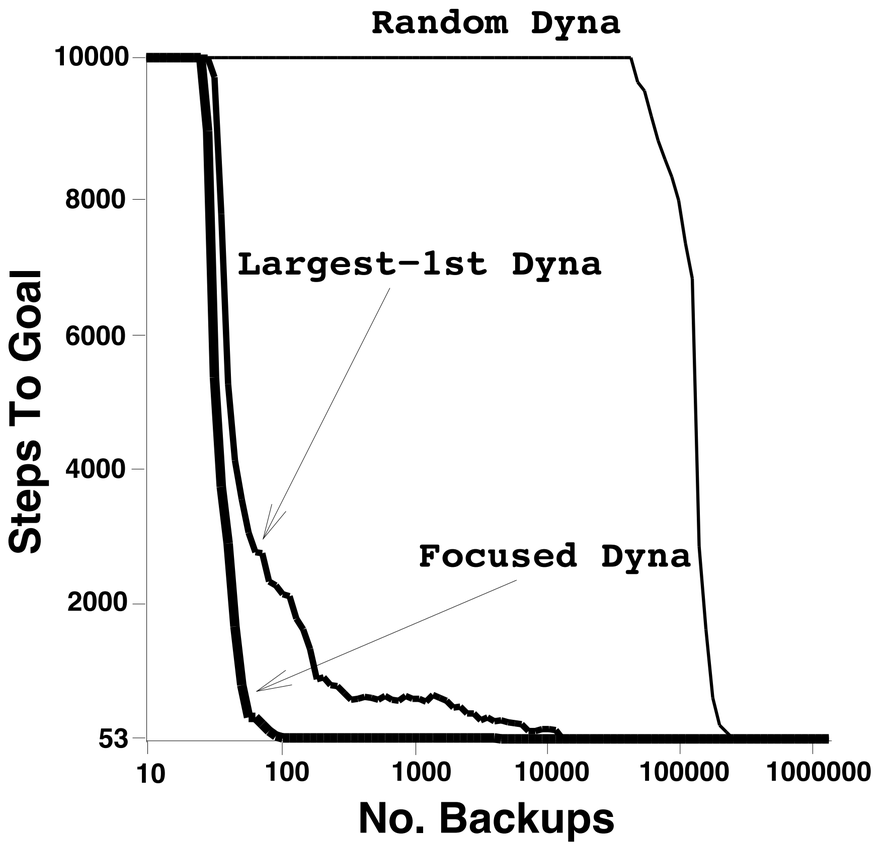
- Consider the second episode in the Dyna maze
  - The agent has successfully reached the goal once…



- In larger problems, the number of states is so large that unfocused planning would be extremely inefficient

Large maze and random search control





current transition, as in Sutton's (1990, 1991) work; (2) queue-Dyna with priority determined by prediction difference magnitude; and (3) queue-Dyna with priority determined by estimated value of the start state. For this last algorithm, estimates of minimum distance from the starting state were maintained using an additional queue, as outlined previously. In the interest of brevity, henceforth we refer to these algorithms as *random-update Dyna*, *largest-first Dyna*, and *focused Dyna*, respectively.

(Peng and Williams, 1993)

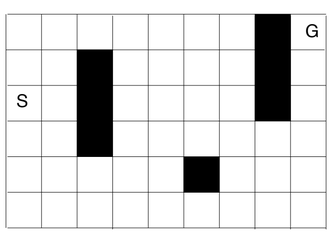
Prioritized Sweeping

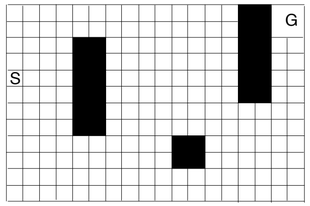
- Which states or state-action pairs should be generated during planning?
- Work backwards from states whose values have just changed:
  - Maintain a queue of state-action pairs whose values would change a lot if backed up, prioritized by the size of the change
  - When a new backup occurs, insert predecessors according to their priorities
  - Always perform backups from first in queue
- Moore & Atkeson 1993; Peng & Williams 1993
- improved by McMahan & Gordon 2005; Van Seijen 2013

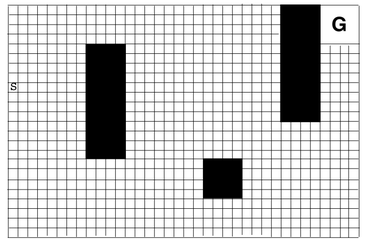
Prioritized Sweeping

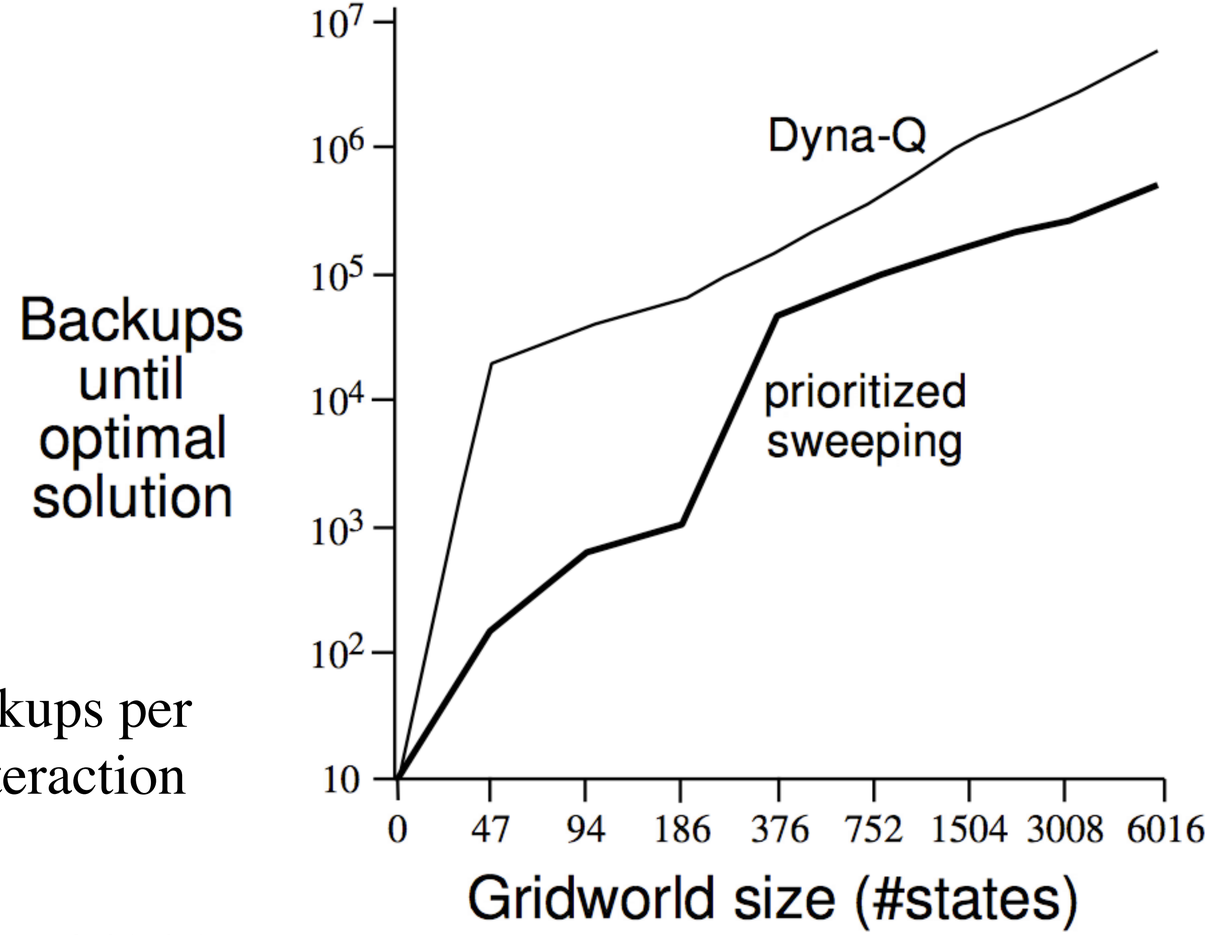
Initialize $Q(s,a)$, $Model(s,a)$, for all s,a, and $PQueue$ to empty
Do forever:
- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow policy(S,Q)$
- (c) Execute action A; observe resultant reward, R, and state, S'
- (d) $Model(S,A) \leftarrow R, S'$
- (e) $P \leftarrow |R + \gamma \max_a Q(S',a) - Q(S,A)|$.
- (f) if $P > \theta$, then insert S,A into $PQueue$ with priority P
- (g) Repeat n times, while $PQueue$ is not empty:
  - $S,A \leftarrow first(PQueue)$
  - $R, S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
  - Repeat, for all $\bar{S}, \bar{A}$ predicted to lead to S:
    - $\bar{R} \leftarrow$ predicted reward for $\bar{S}, \bar{A}, S$
    - $P \leftarrow |\bar{R} + \gamma \max_a Q(S,a) - Q(\bar{S},\bar{A})|$.
    - if $P > \theta$ then insert $\bar{S}, \bar{A}$ into $PQueue$ with priority P

Prioritized Sweeping vs. Dyna-Q



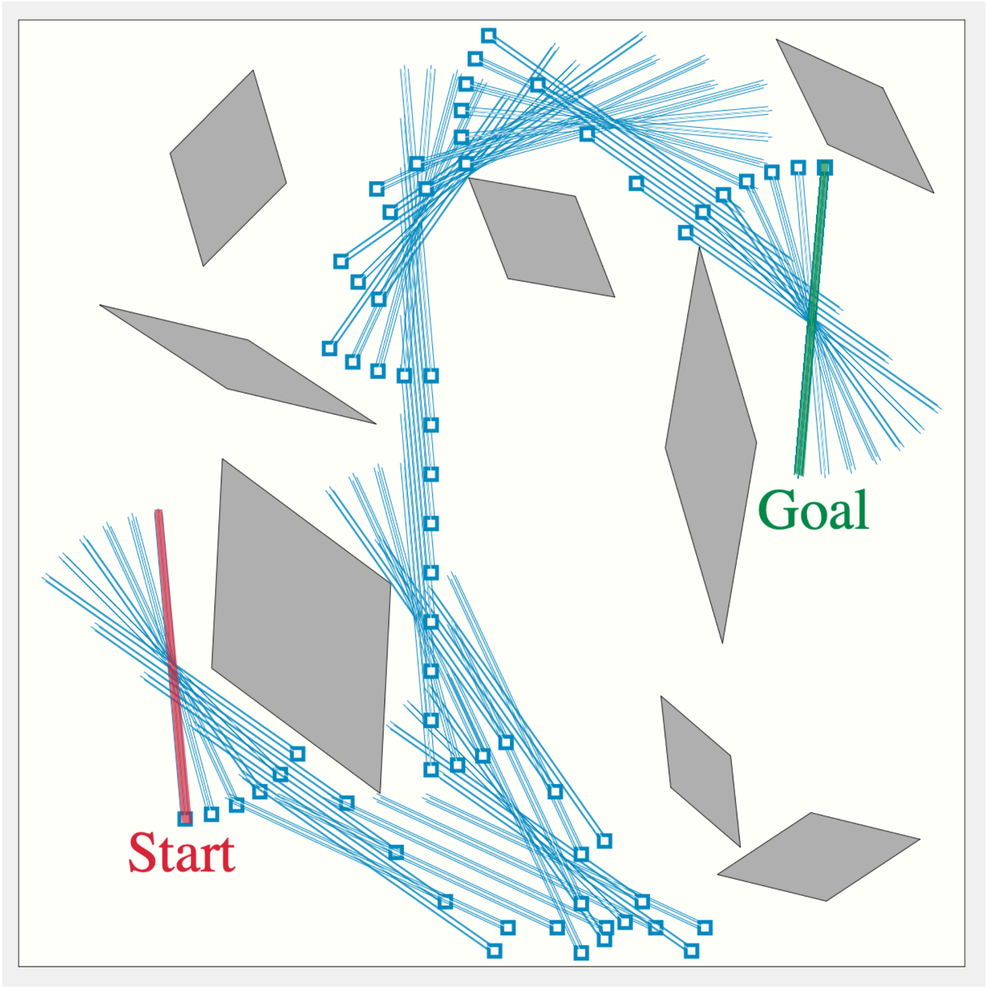






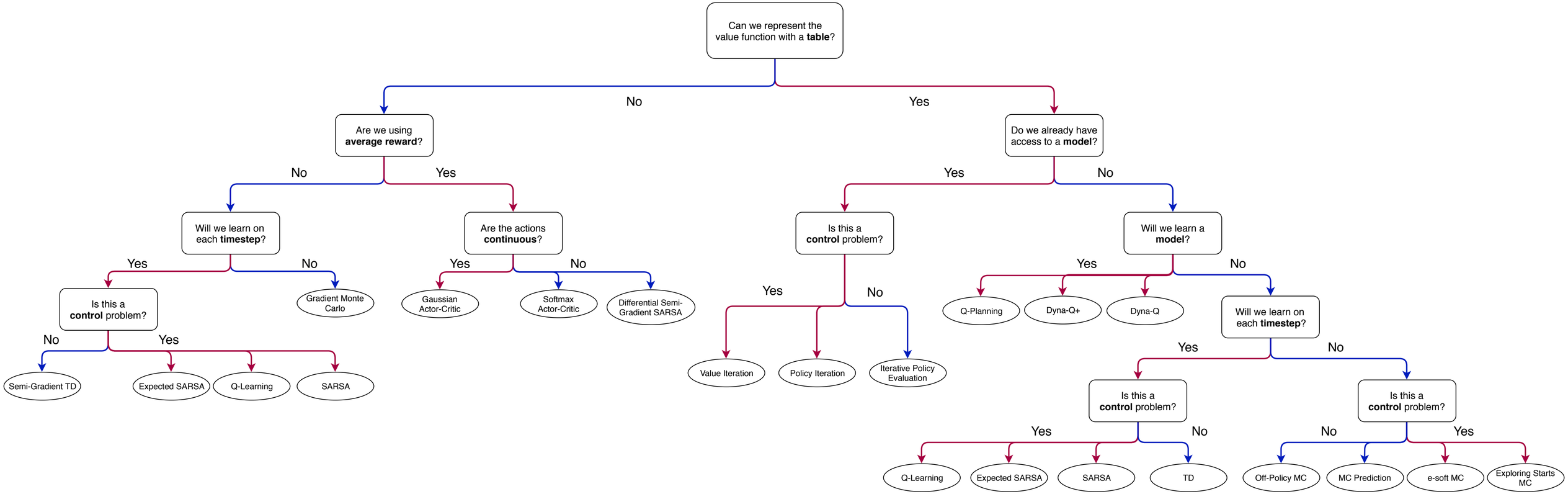
Both use n=5 backups per environmental interaction

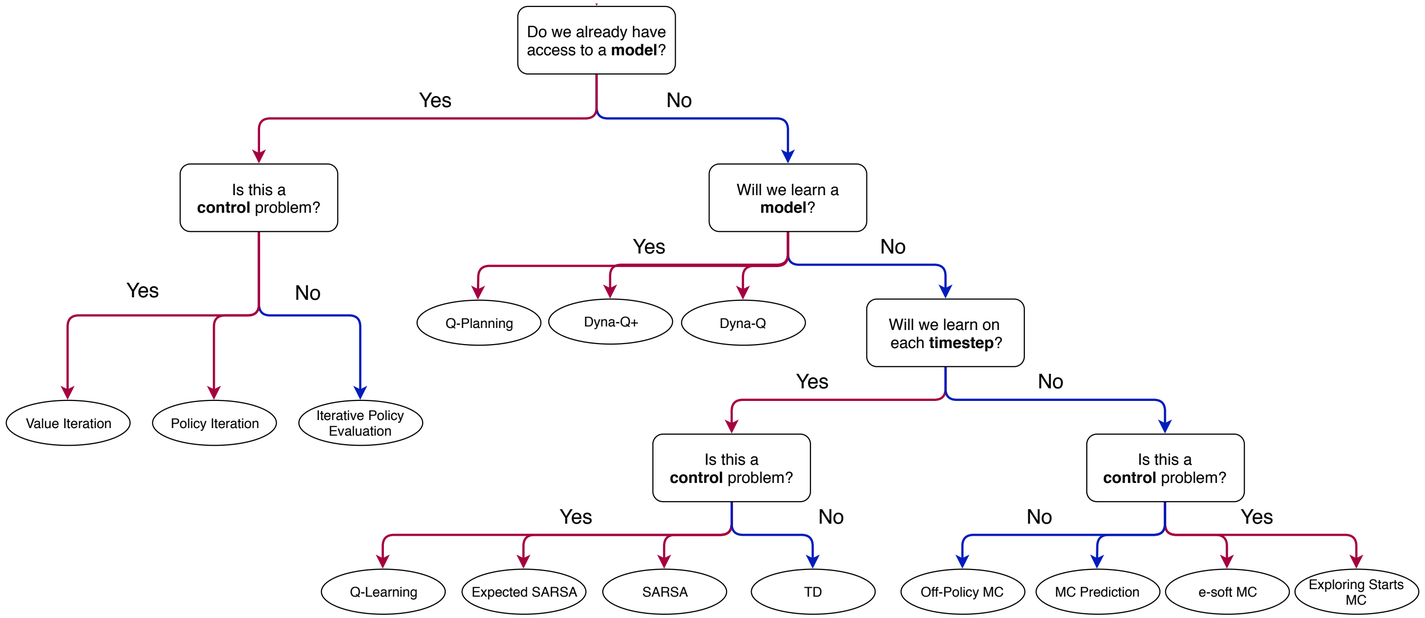
Rod Maneuvering (Moore and Atkeson 1993)

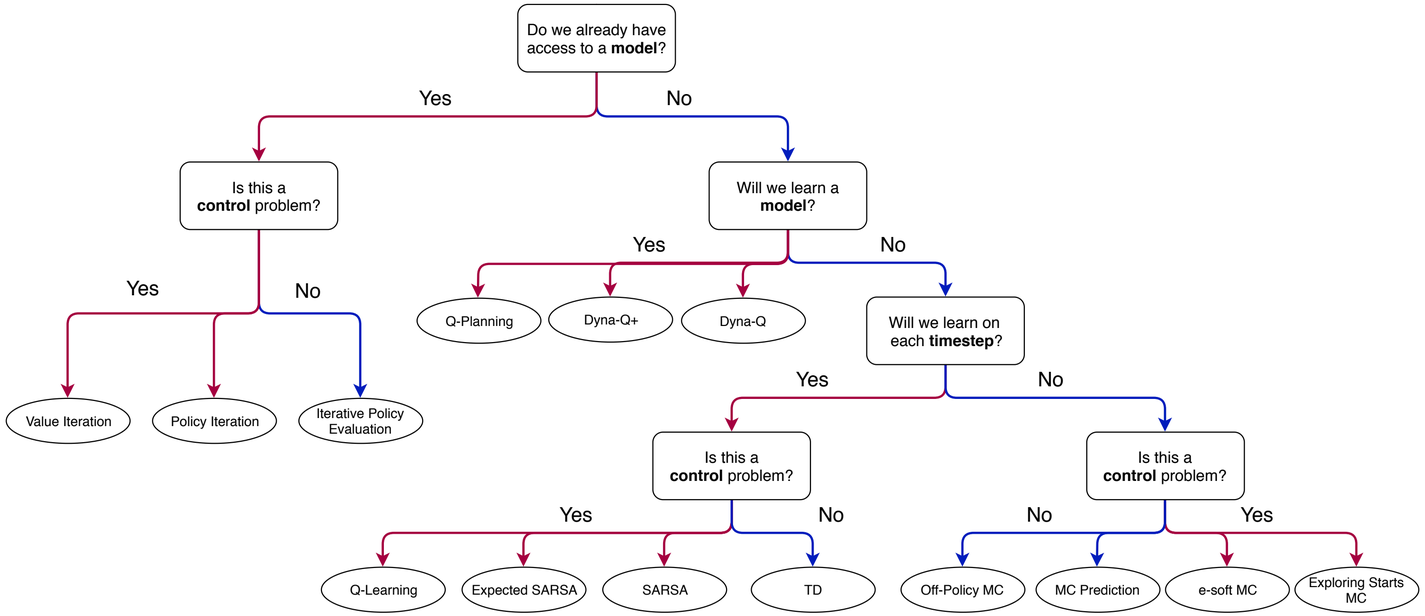


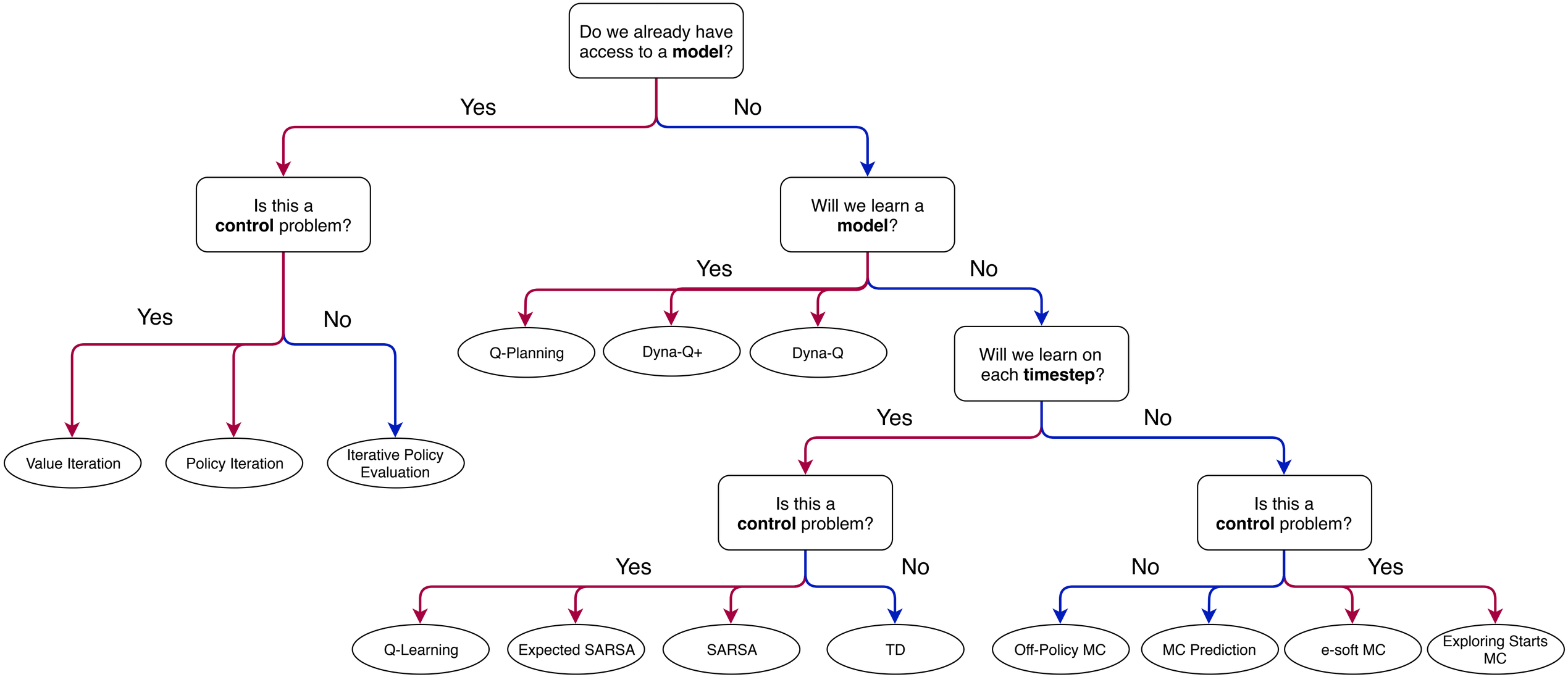
So many algorithms! What is a researcher to do?

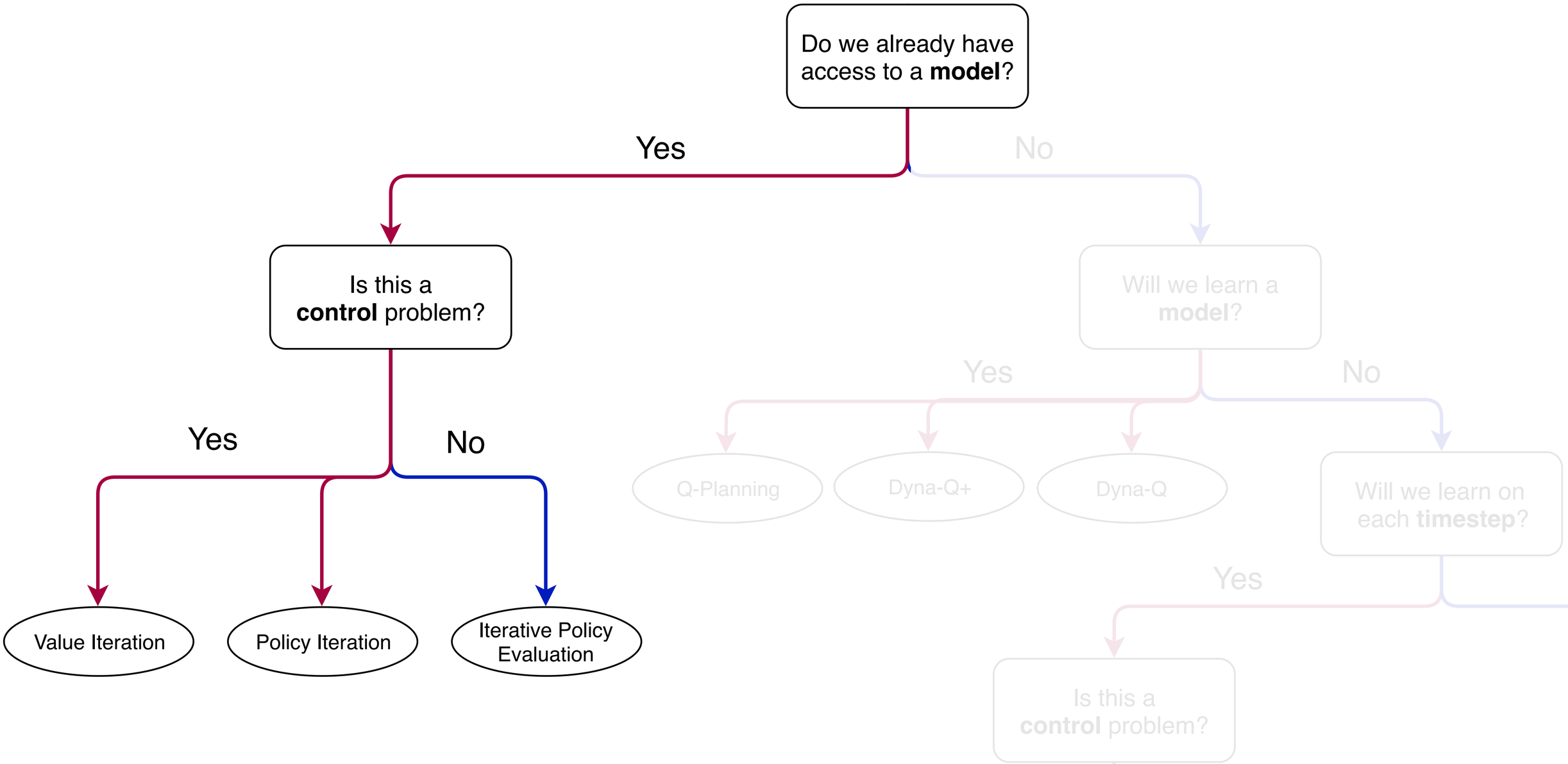
- Introducing the **Course Map**

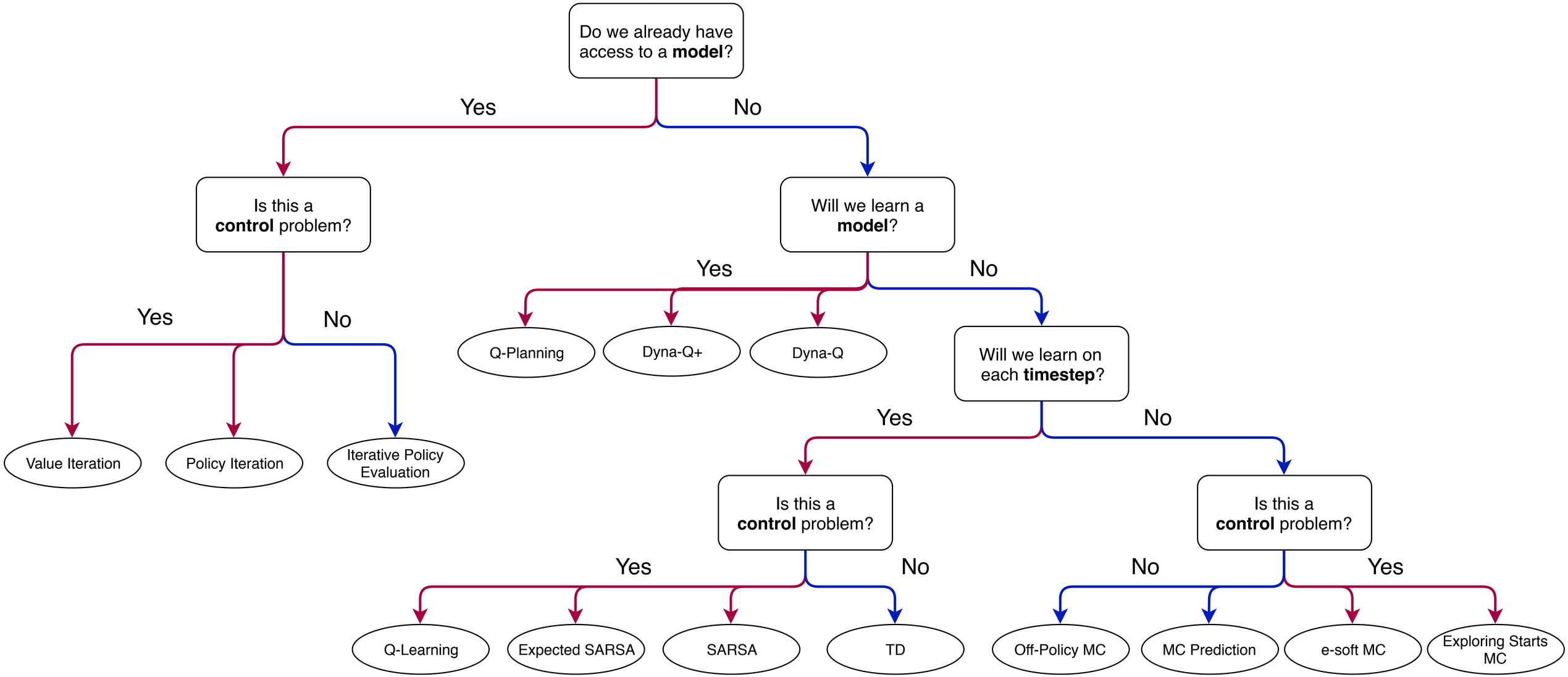

Can we represent the value function with a **table**?
No
Yes
Are we using **average reward**?
Do we already have access to a **model**?
No
Yes
Yes
No
Will we learn on each **timestep**?
Are the actions **continuous**?
Is this a **control** problem?
Will we learn a **model**?
Yes
No
Yes
No
Yes
No
Yes
No
Is this a **control** problem?
Gradient Monte Carlo
Gaussian Actor-Critic
Softmax Actor-Critic
Differential Semi-Gradient SARSA
Value Iteration
Policy Iteration
Iterative Policy Evaluation
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
No
Yes
Yes
No
Yes
No
Semi-Gradient TD
Expected SARSA
Q-Learning
SARSA
Is this a **control** problem?
Is this a **control** problem?
Yes
No
No
Yes
Q-Learning
Expected SARSA
SARSA
TD
Off-Policy MC
MC Prediction
e-soft MC
Exploring Starts MC

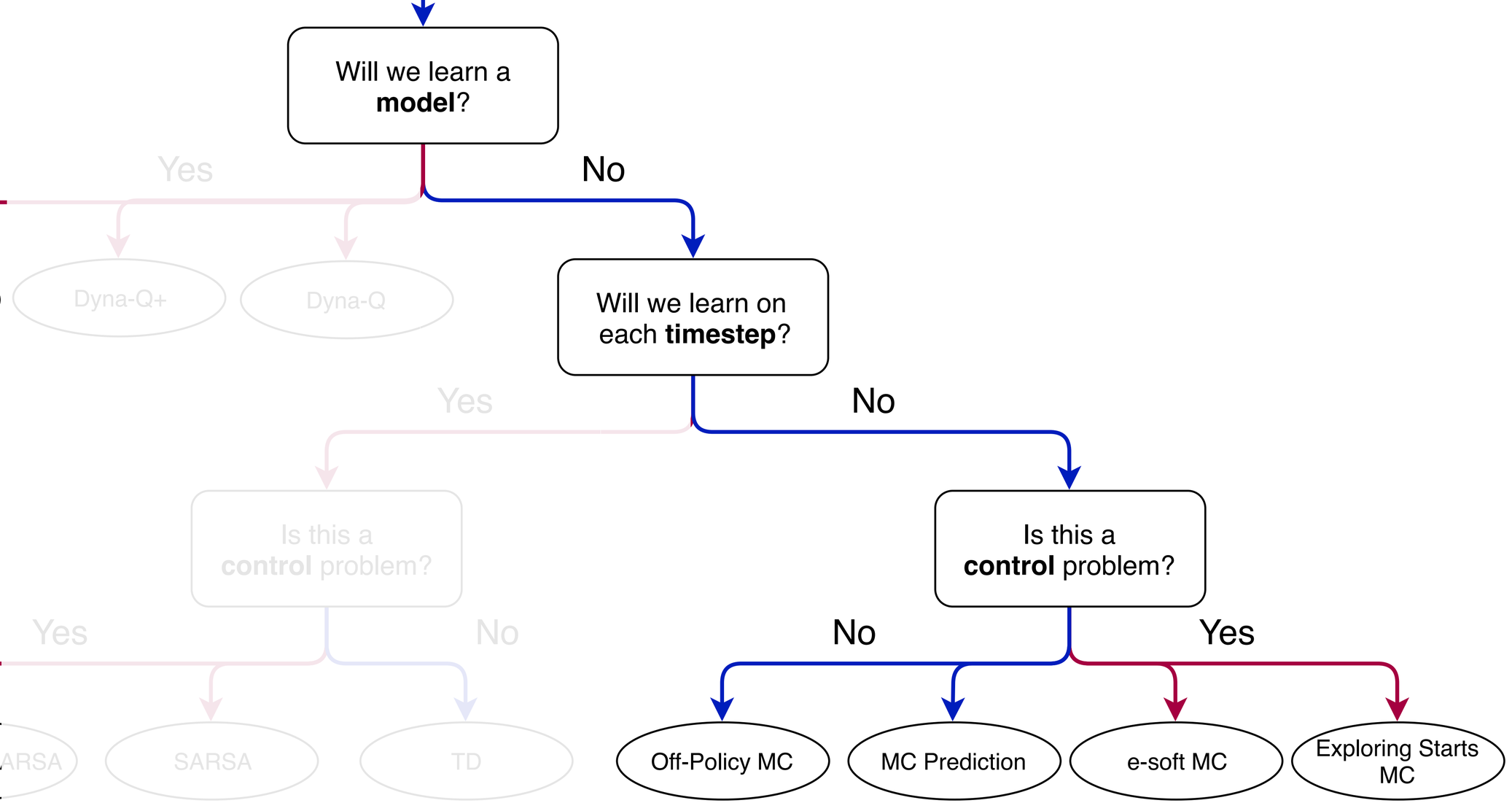

Do we already have access to a **model**?
Yes
No
Is this a **control** problem?
Will we learn a **model**?
Yes
No
Yes
No
Value Iteration
Policy Iteration
Iterative Policy Evaluation
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Yes
No
Is this a **control** problem?
Is this a **control** problem?
Yes
No
No
Yes
Q-Learning
Expected SARSA
SARSA
TD
Off-Policy MC
MC Prediction
e-soft MC
Exploring Starts MC

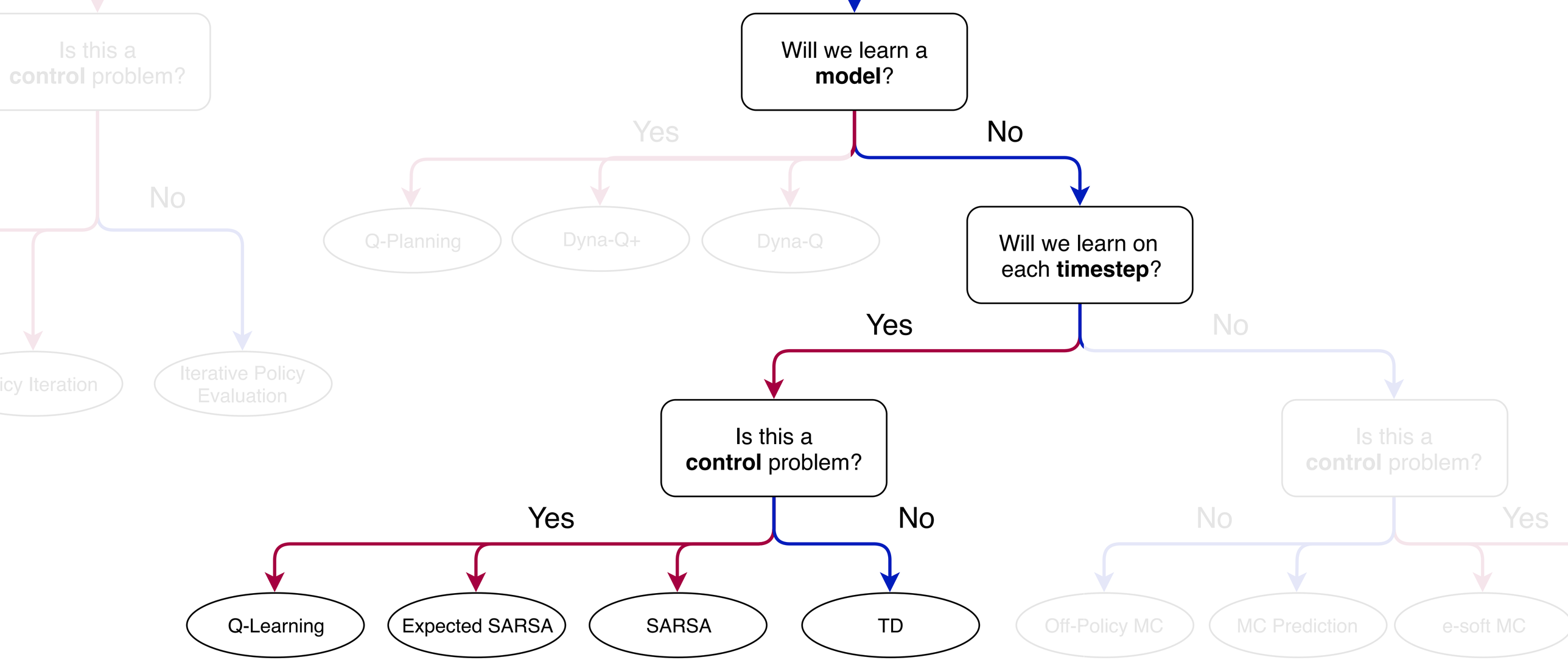

Do we already have access to a **model**?
Yes
No
Is this a **control** problem?
Will we learn a **model**?
Yes
No
Yes
No
Value Iteration
Policy Iteration
Iterative Policy Evaluation
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Yes
No
Is this a **control** problem?
Is this a **control** problem?
Yes
No
No
Yes
Q-Learning
Expected SARSA
SARSA
TD
Off-Policy MC
MC Prediction
e-soft MC
Exploring Starts MC

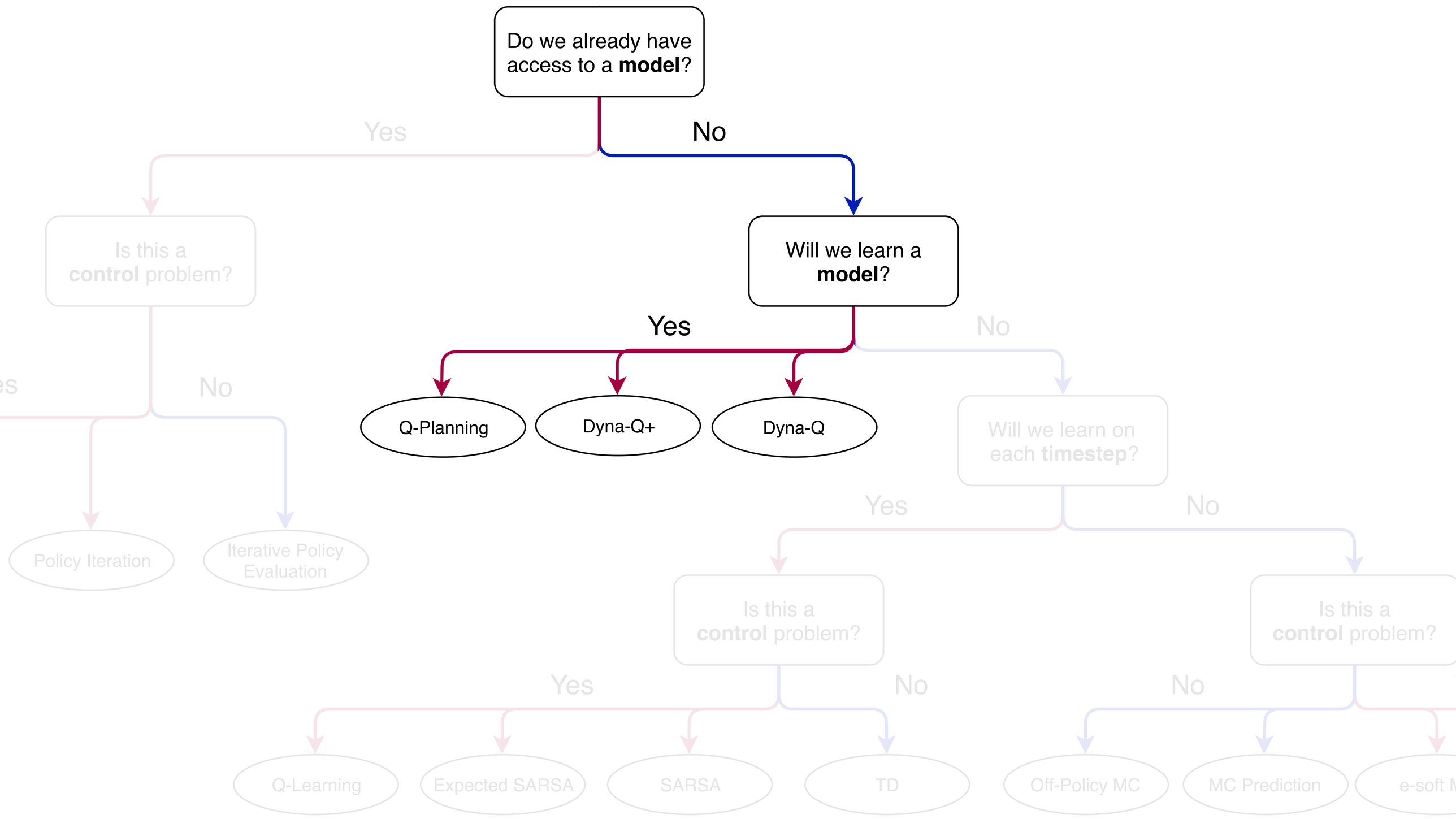

Do we already have access to a **model**?
Yes
No
Is this a **control** problem?
Will we learn a **model**?
Yes
No
Yes
No
Value Iteration
Policy Iteration
Iterative Policy Evaluation
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Yes
No
Is this a **control** problem?
Is this a **control** problem?
Yes
No
No
Yes
Q-Learning
Expected SARSA
SARSA
TD
Off-Policy MC
MC Prediction
e-soft MC
Exploring Starts MC


Do we already have access to a **model**?
Yes
No
Is this a **control** problem?
Will we learn a **model**?
Yes
No
Yes
No
Value Iteration
Policy Iteration
Iterative Policy Evaluation
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Yes
Is this a **control** problem?


Do we already have access to a **model**?
Yes
No
Is this a **control** problem?
Will we learn a **model**?
Yes
No
Yes
No
Value Iteration
Policy Iteration
Iterative Policy Evaluation
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Yes
No
Is this a **control** problem?
Is this a **control** problem?
Yes
No
No
Yes
Q-Learning
Expected SARSA
SARSA
TD
Off-Policy MC
MC Prediction
e-soft MC
Exploring Starts MC


Will we learn a **model**?
Yes
No
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Yes
No
Is this a **control** problem?
Yes
No
SARSA
TD
Is this a **control** problem?
No
Yes
Off-Policy MC
MC Prediction
e-soft MC
Exploring Starts MC


Will we learn a **model**?
Yes
No
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Yes
No
Is this a **control** problem?
Yes
No
Q-Learning
Expected SARSA
SARSA
TD
Is this a **control** problem?
No
Yes
Off-Policy MC
MC Prediction
e-soft MC
Is this a **control** problem?
No
Yes
Iterative Policy Evaluation
icy Iteration


Do we already have access to a **model**?
Yes
No
Is this a **control** problem?
Will we learn a **model**?
Yes
No
es
No
Q-Planning
Dyna-Q+
Dyna-Q
Will we learn on each **timestep**?
Policy Iteration
Iterative Policy Evaluation
Yes
No
Is this a **control** problem?
Is this a **control** problem?
Yes
No
No
Q-Learning
Expected SARSA
SARSA
TD
Off-Policy MC
MC Prediction
e-soft