Course 1, Module 5
Dynamic Programming

CMPUT 397
Fall 2019

Your feedback (google survey results)

- Cover key terminology during review session
- Practice questions were a big positive
- Some people wanted more challenging in-class questions
- Some people wanted less discussion time
- Some people wanted more ways to ask questions

- We will do another survey in a couple weeks!

Reminders: Sept 18, 2019

- **Dynamic Programming Course 1, Module 4 (Notebook) due this friday**
- The private sessions for Course 2 (Sample based learning methods) is ready
  - link to private session: https://coursera.org/groups/sample-based-learning-methods-ianmk/invitation (see eclass for more detailed instructions)
- We start Course 2 material next week!

New format for Wednesday Lecture

- Based on your feedback
- You will still submit weekly discussion questions
- Wednesday lecture will begin with clarifying misconceptions
- The last 15-20 minutes of lecture we will break into **three groups**:

1. **Q&A** with the **TAs**: ask them anything about the course (goto them)
2. **Challenge questions** with **Martha**
3. **Discussion** time with **Adam**

Live question submission in class

- goto: https://www.tricider.com/brainstorming/3ZFvVmeuw4N
- Submit a question (anonymous if you like)
- We will cover them live in lecture today
    - and if it works we will do this every Wednesday

Clarifications

- How can we **scale DP**?
- When would DP **not be effective**?
- Why is the value of the **terminal** state always **zero?**
- Why would we **initialise** the policy/value function different from zero? (What are the impacts on initialisation)?
- Why is there no epsilon-greedy (**exploration**) in Dynamic Programming
- Does DP only work with **deterministic policies**?

Clarifications

- Why is the **stopping condition** in policy evaluation max(delta ,|v - V (s)|), **not** min(delta ,|v - V (s)|)
- Statements about **convergence** can be different than statements about how the algorithm works in **practice** (note we stop the algorithms before infinity)
- In Policy Iteration, the policy evaluation step stops based on **theta**, can we say anything about **convergence given different values of theta (Martha)**
- **The policy improvement theorem states we greedify the policy, how does this not result in local minimum?**
- If there are multiple optimal policies, policy iteration will always return the first one. What do we do about this? **(Challenge question)**

Clarifications

- Why is **bootstrapping** required in DP? When would bootstrapping be a bad idea?
- Are there situations where we would use the **Monte Carlo method instead of DP**?
- How do we use **Monte Carlo** for policy **evaluation**?
- How does DP work in **changing environments**? Think of a maze where a path gets blocked off.
- What is better Policy Iteration or Value Iteration? (Historical Remarks)
- Can we really apply DP? **Warren Powell: Approximate Dynamic Programming for Fleet Management**

Discussion topics for today

1. Why might **policy** iteration be **better** than **value** iteration? Or the other way around?

2. How do we decide **which states** to update in **asynchronous** DP?

3. How could we **apply** DP methods in real-world **problems** with large (even infinite state-spaces)

Challenge Question 1

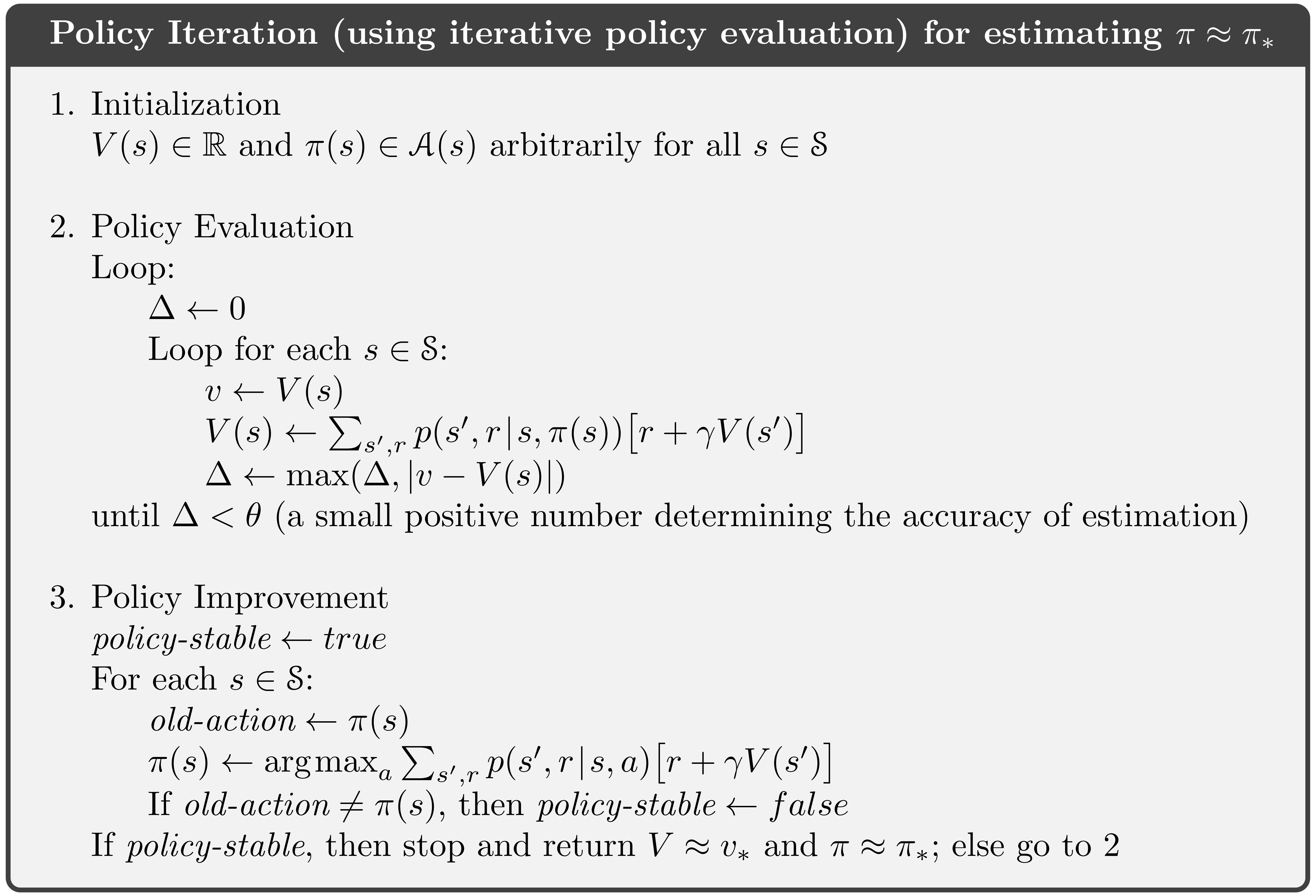
4. **(Challenge Question)** A gambler has the opportunity to make bets on the outcomes of a sequence of coin flips. If the coin comes up heads, she wins as many dollars as she has staked on that flip; if it is tails, she loses her stake. The game ends when the gambler wins by reaching her goal of \$100, or loses by running out of money. On each flip, the gambler must decide what portion of her capital to stake, in integer numbers of dollars. This problem can be formulated as an undiscounted, episodic, finite MDP. The state is the gambler's capital, $s \in \{1, 2, ..., 99\}$ and the actions are stakes, $a \in \{0, 1, ..., \min(s, 100 - s)\}$. The reward is $+1$ when reaching the goal of \$100 is zero on all other transitions.

   (a) What is the interpretation for the value function in this problem?

   (b) Write down pseudocode to solve this problem. Think about how you can modify improve efficiency of the Policy Iteration algorithm for this problem, by exploiting your knowledge of the dynamics.

Challenge Question 2

5. (*Exercise 4.4 S&B*) The policy iteration algorithm on page 80 has a subtle bug in that it may never terminate if the policy continually switches between two or more policies that are equally good. This is ok for pedagogy, but not for actual use. Modify the pseudocode so that convergence is guaranteed. Note that there is more than one approach to solve this problem.

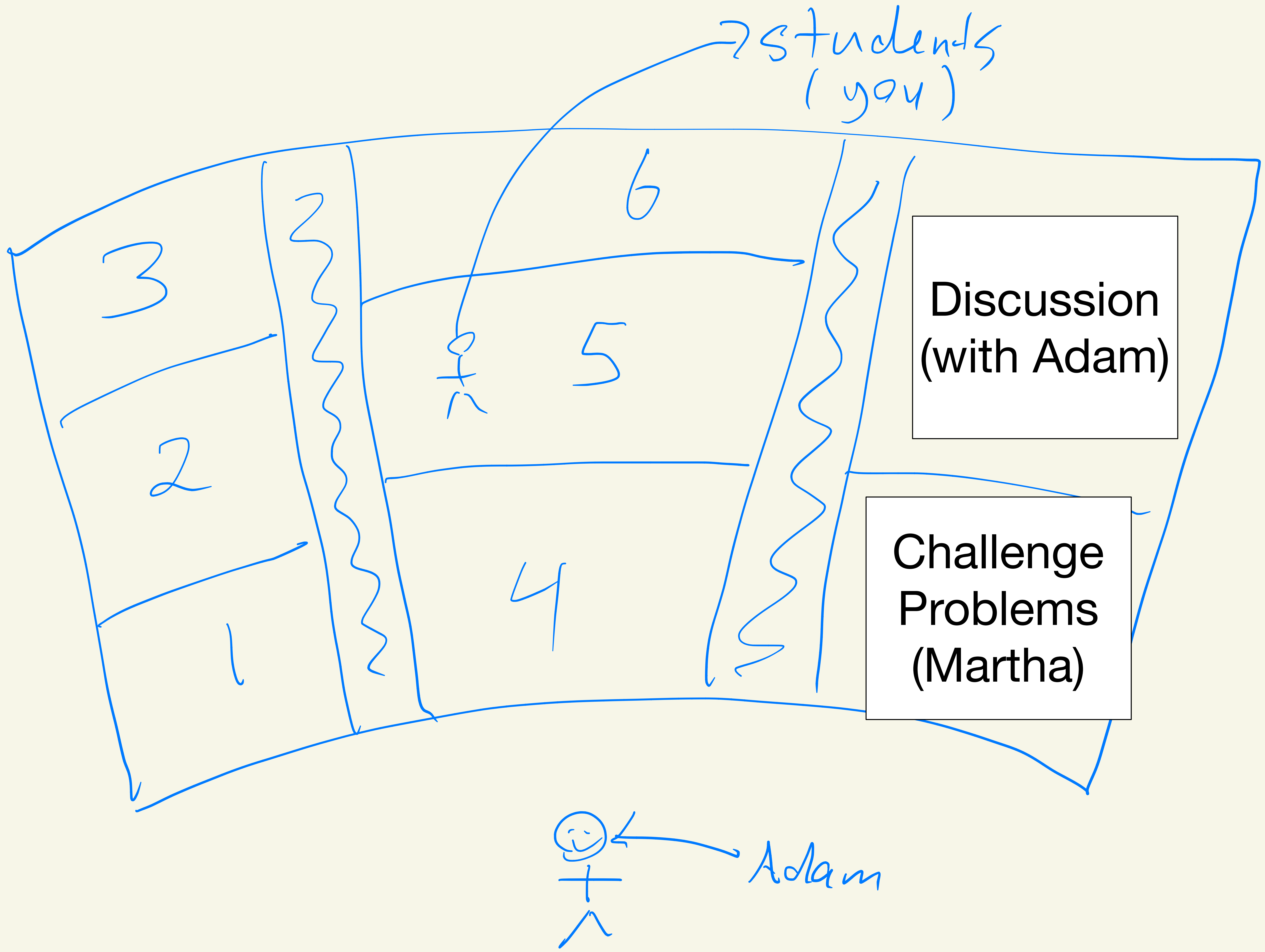


Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
       $\Delta \leftarrow 0$
       Loop for each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       *old-action* $\leftarrow \pi(s)$
       $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma V(s')\big]$
       If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ $false$
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```



Discussion (with Adam)

Challenge Problems (Martha)