Markov Decision Processes

Chen-Yu Wei

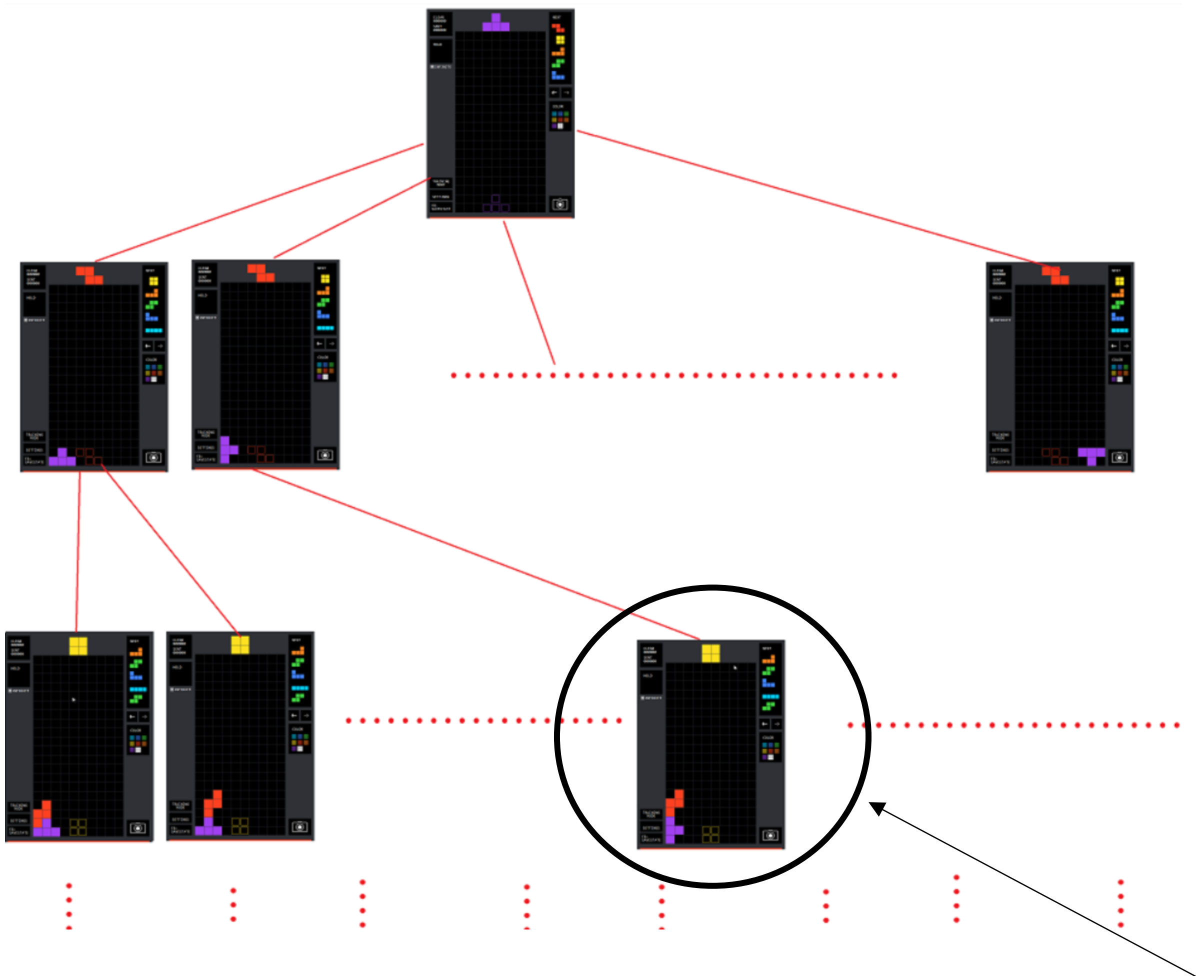
Sequence of Actions

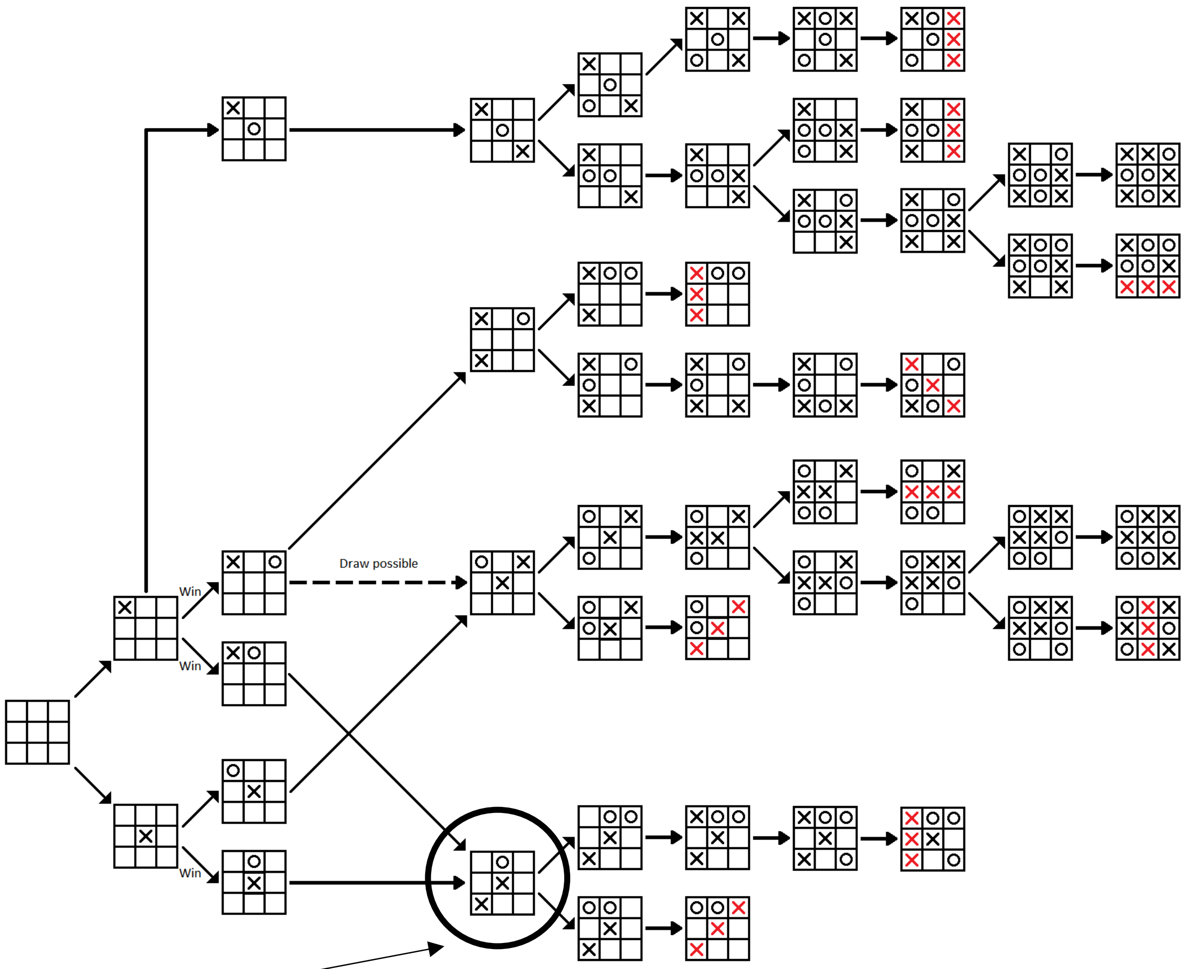
To win the game, the learner has to take a sequence of actions $a_1 \to a_2 \to \cdots \to a_H$.

The effect of a particular action may not be revealed instantaneously.

- Some effect may be revealed instantaneously
- Some may be revealed later

Sequence of Actions





state

(a summary of the current status in a multi-stage game)

Deterministic World

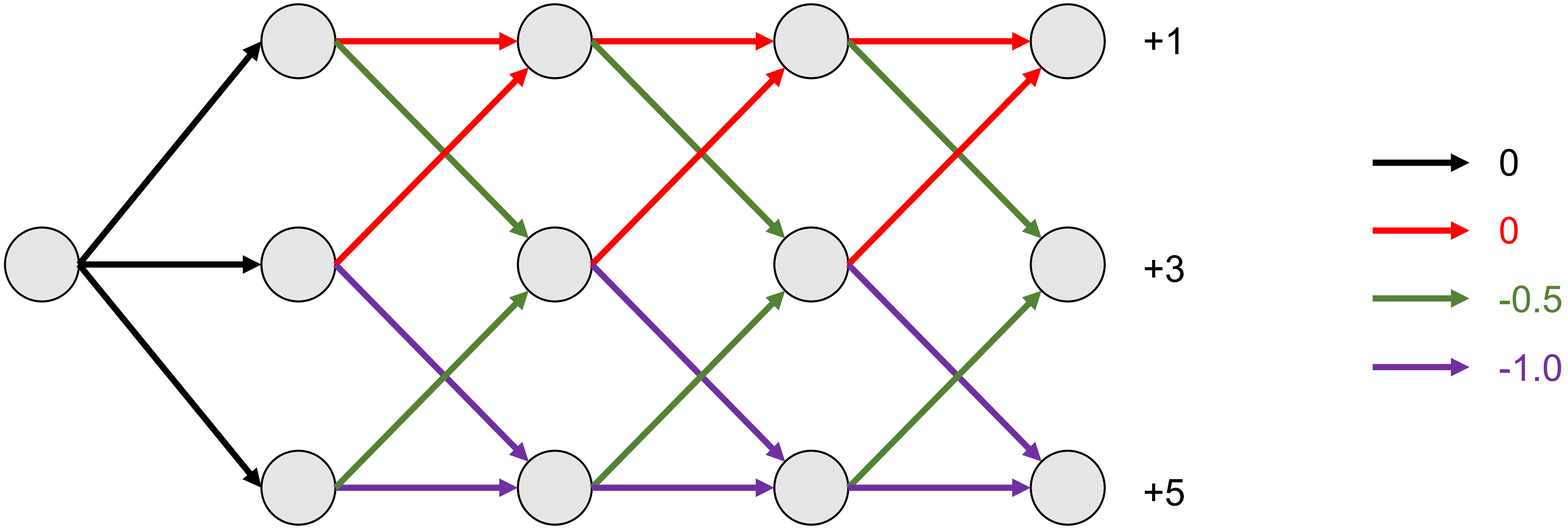
$$R(a)$$

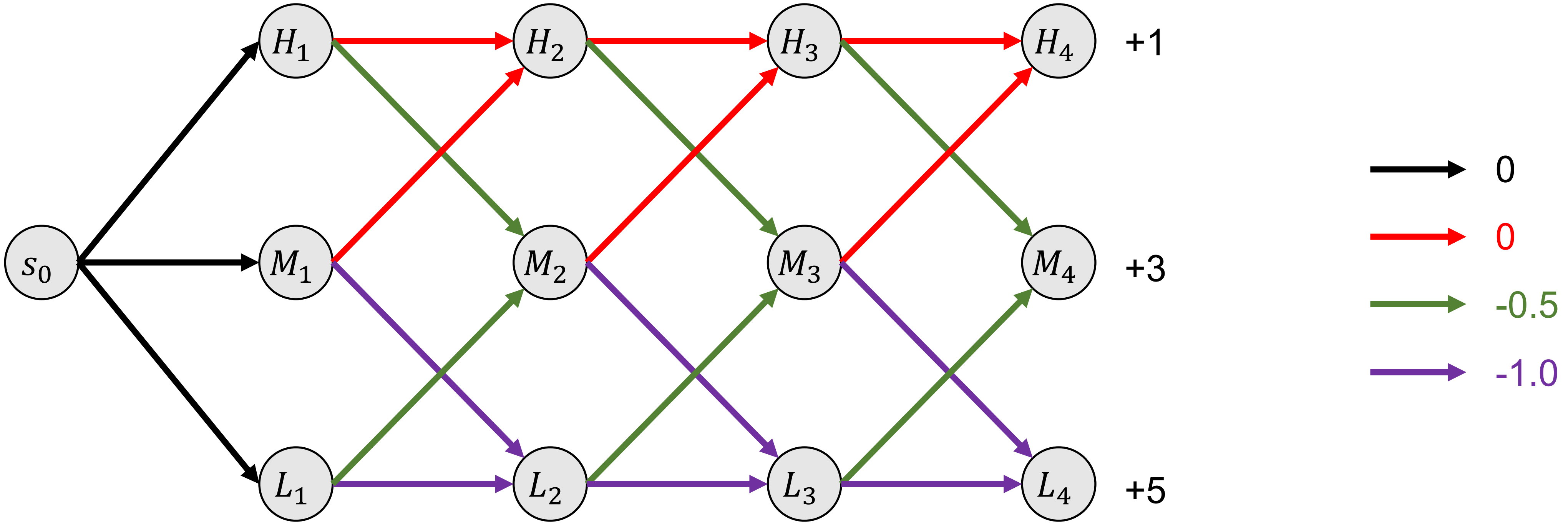
$$R(x,a)$$

$$\pi^* = \operatorname*{argmax}_a R(a)$$

$$\pi^*(x) = \operatorname*{argmax}_a R(x,a)$$

Which path gives us the highest **total reward**?

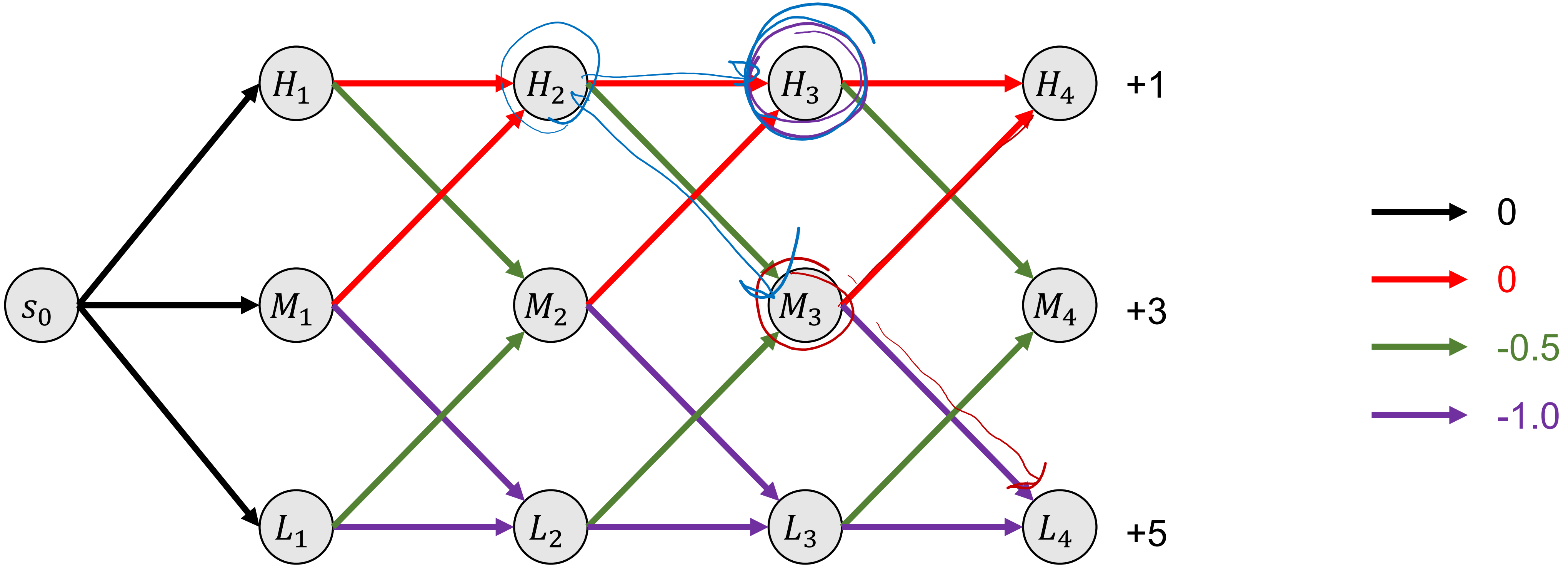




$V^\star(s)$ ≔ maximum total reward starting from state s

$Q^\star(s, a)$ ≔ maximum total reward starting from state s and taking action a **for one step**, and then following the optimal strategy

$\pi^\star(s)$ ≔ optimal decision on state s



$V^\star(H_4) =$ 1 $\quad Q^\star(H_3, R) =$ 0+1=1 $\quad V^\star(H_3) =$ 2.5 $\quad Q^\star(H_2, R) =$ 0+2.5=2.5 $\quad V^\star(H_2) =$ 3.5

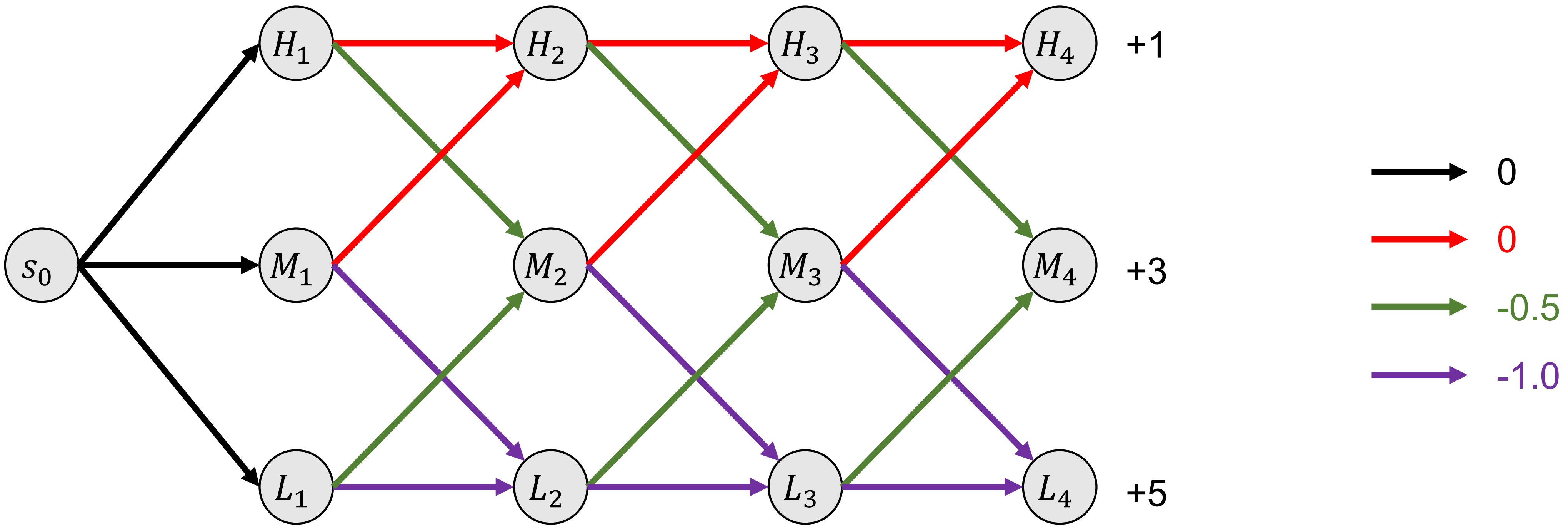
$\boxed{Q^\star(H_3, G)} =$ -0.5+3=2.5 $\quad \boxed{Q^\star(H_2, G)} =$ -0.5+4=3.5

$V^\star(M_4) =$ 3 $\quad Q^\star(M_3, R) =$ 0+1=1 $\quad V^\star(M_3) =$ 4 $\quad Q^\star(M_2, R) =$ $\quad V^\star(M_2) =$?

$\boxed{Q^\star(M_3, P)} =$ -1+5=4 $\quad Q^\star(M_2, P) =$

$V^\star(L_4) =$ 5 $\quad Q^\star(L_3, G) =$ -0.5+3=2.5 $\quad V^\star(L_3) =$ 4 $\quad Q^\star(L_2, G) =$ $\quad V^\star(L_2) =$?

$\boxed{Q^\star(L_3, P)} =$ -1+5=4 $\quad Q^\star(L_2, P) =$

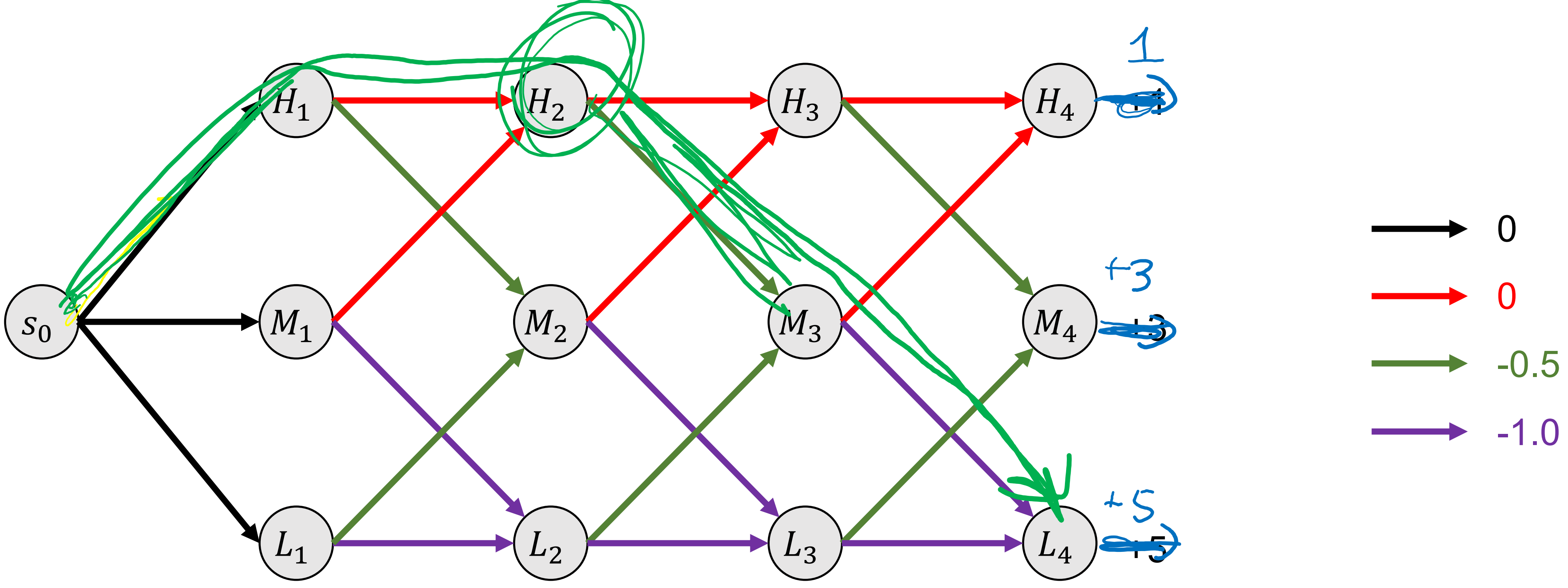


Relation between $Q^\star, V^\star, \pi^\star$:

$$Q^\star(s,a) = R(s,a) + V^\star(\text{next\_state}(s,a))$$

$$V^\star(s) = \max_a Q^\star(s,a)$$

$$\pi^\star(s) = \operatorname*{argmax}_a Q^\star(s,a)$$



General algorithm (Value Iteration) --- see next slide for a more detailed version

```
Repeat until Q, V no longer changes:
        $Q(s,a) \leftarrow R(s,a) + V(\text{next\_state}(s,a))$    for all $(s,a)$
        $V(s) = \max_a Q(s,a)$                                      for all $s$
$\pi^\star(s) = \operatorname{argmax}_a Q(s,a)$
```

(for the last layer states. $Q(s,a) \leftarrow R(s,a)$)

Value Iteration:

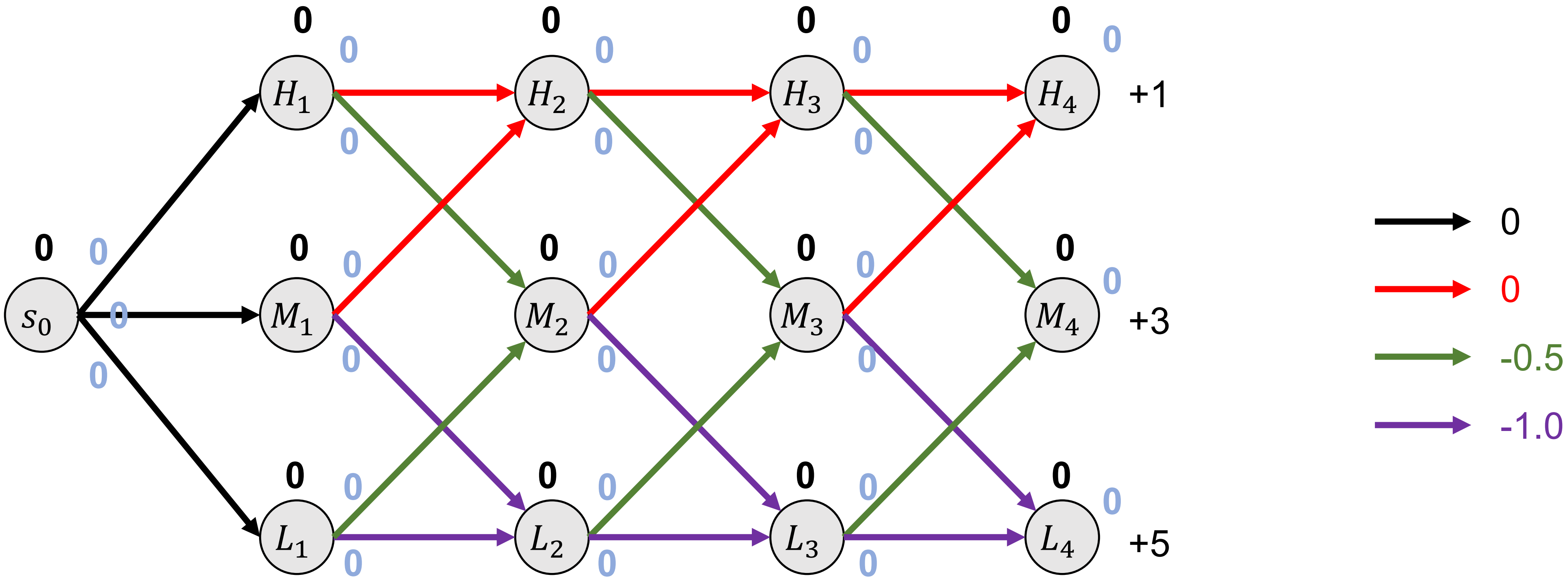
```
Initialize $Q_0(s,a) \leftarrow 0,\ V_0(s) \leftarrow 0$ for all $(s,a)$
For $i = 1, 2, \ldots$
    $Q_i(s,a) \leftarrow R(s,a) + V_{i-1}(\text{next\_state}(s,a))$ for all $(s,a)$
    $V_i(s) = \max_a Q_i(s,a)$ for all $s$
    If $Q_i(s,a) = Q_{i-1}(s,a)$ for all $(s,a)$: **break**
$\pi^\star(s) = \operatorname{argmax}_a Q_{\text{final}}(s,a)$
```

If s is a terminal state, this line is simply $Q_i(s,a) \leftarrow R(s,a)$

If every path in the graph has length $\leq K$, then Value Iteration will terminate in $\leq K + 1$ iterations.

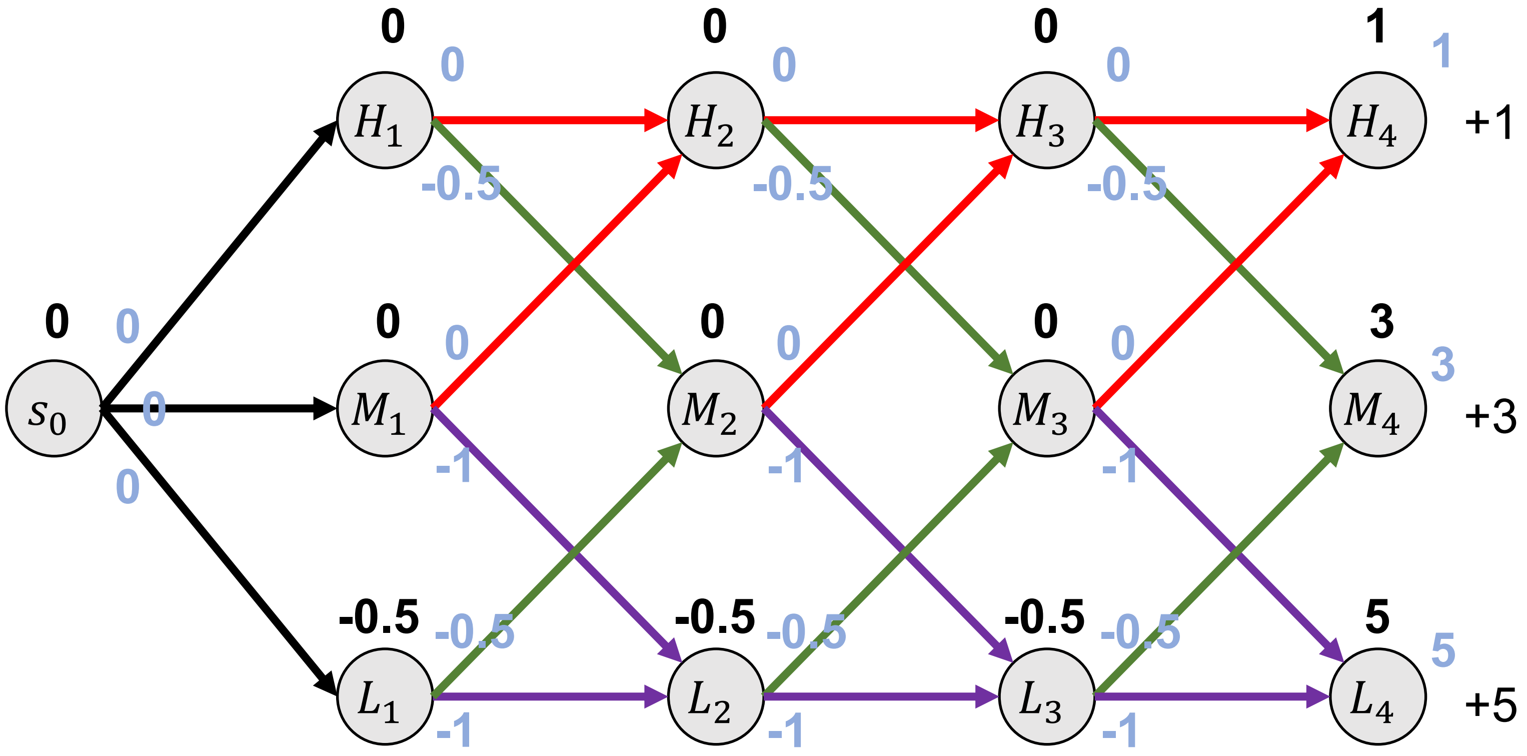
$$Q_0(s, a) = 0$$

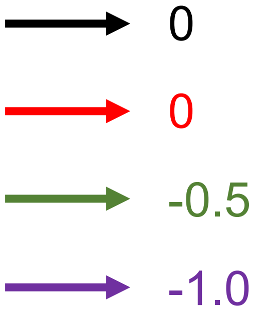
$$V_0(s) = 0$$


0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
s_0
H_1
M_1
L_1
H_2
M_2
L_2
H_3
M_3
L_3
H_4
M_4
L_4
+1
+3
+5
0
0
-0.5
-1.0

$$Q_1(s,a) = R(s,a) + V_0(\text{next}(s,a))$$

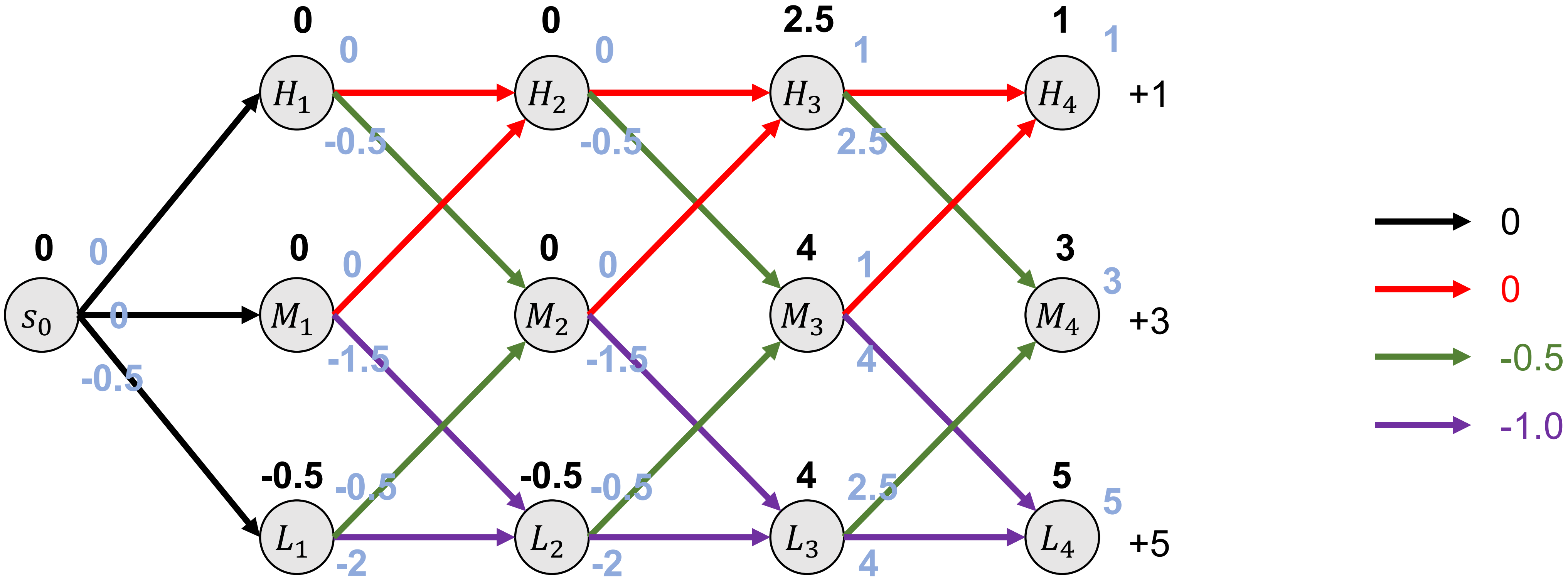
$$V_1(s) = \max_a Q_1(s,a)$$





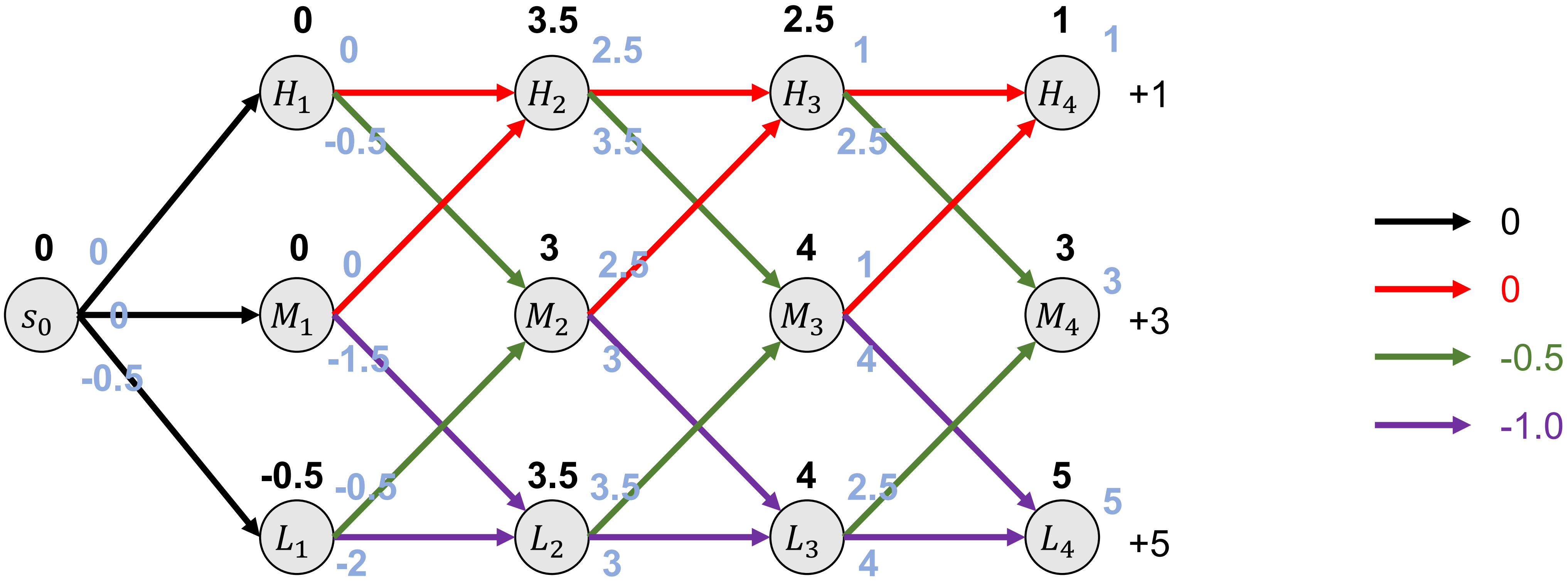
$$Q_2(s,a) = R(s,a) + V_1(\text{next}(s,a))$$

$$V_2(s) = \max_a Q_2(s,a)$$



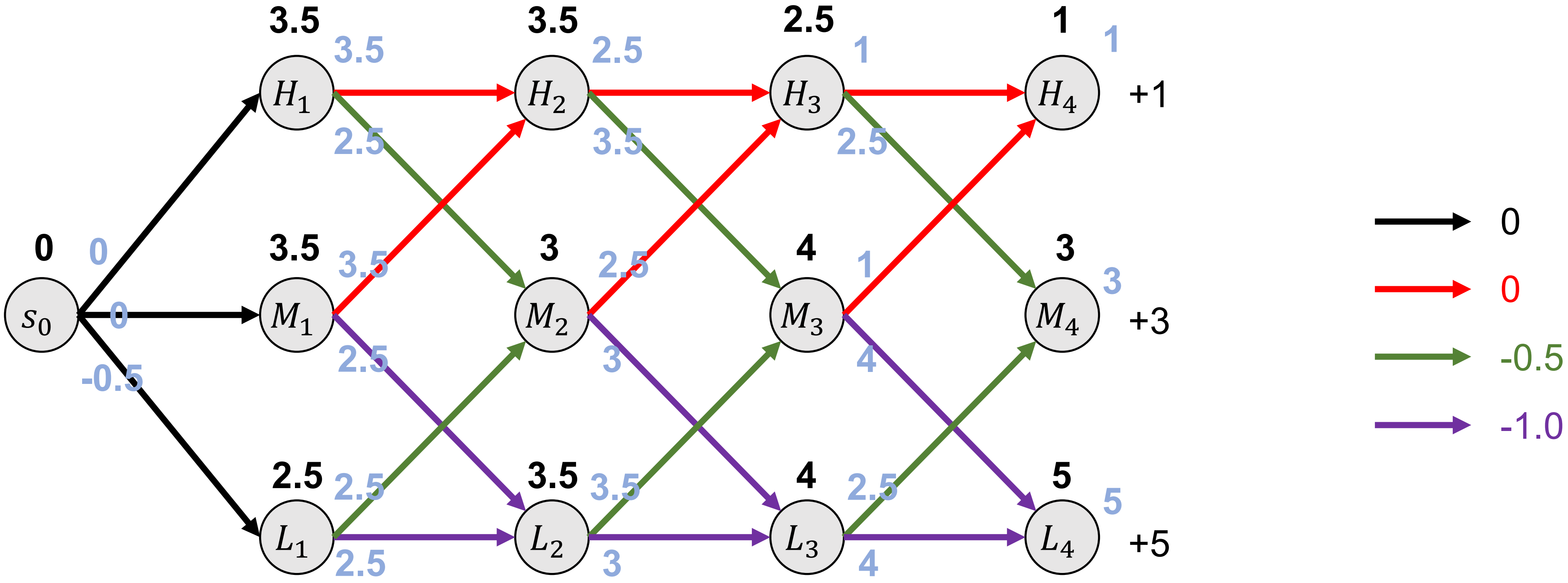
$$Q_3(s,a) = R(s,a) + V_2(\text{next}(s,a))$$

$$V_3(s) = \max_a Q_3(s,a)$$



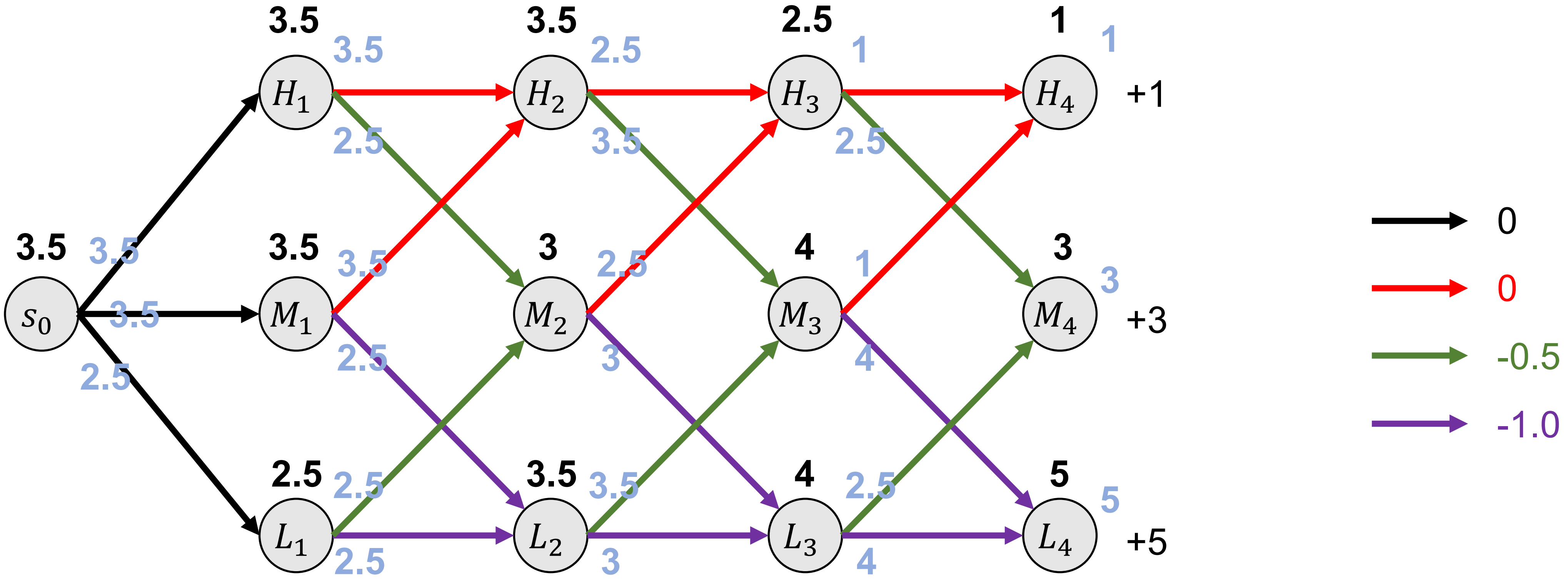
$$Q_4(s,a) = R(s,a) + V_3(\text{next}(s,a))$$

$$V_4(s) = \max_a Q_4(s,a)$$



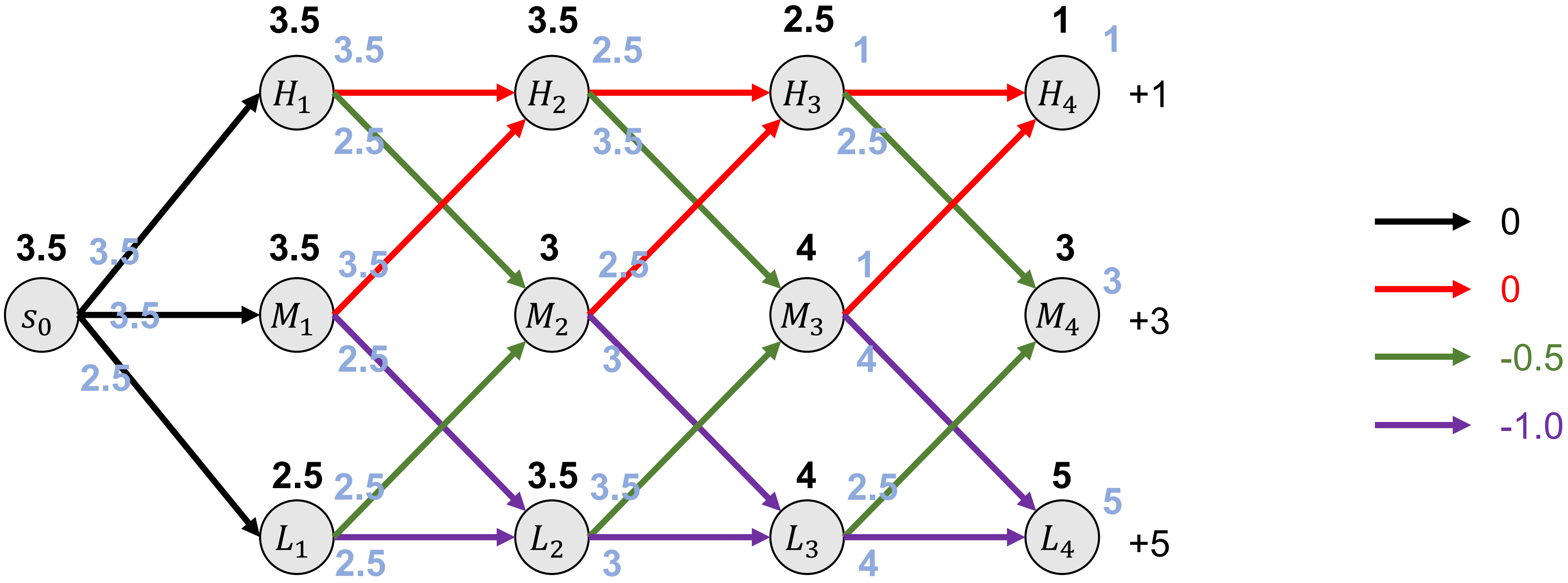
$$Q_5(s,a) = R(s,a) + V_4(\mathrm{next}(s,a))$$

$$V_5(s) = \max_a Q_5(s,a)$$



$$Q_6(s,a) = R(s,a) + V_5(\text{next}(s,a))$$

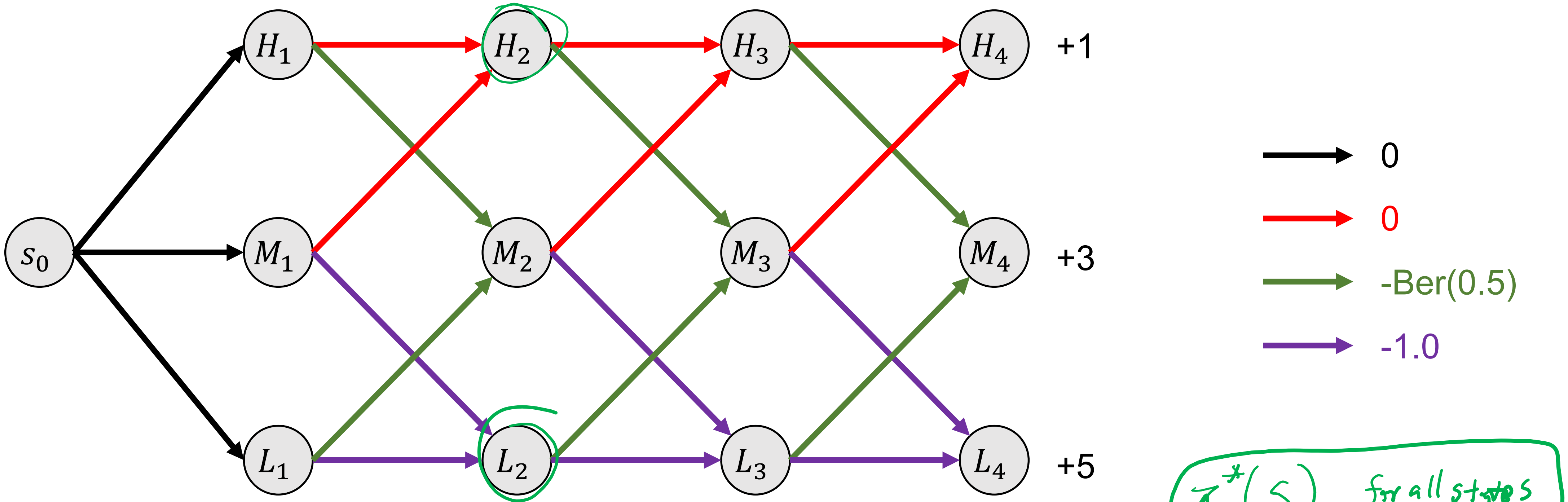
$$V_6(s) = \max_a Q_6(s,a)$$



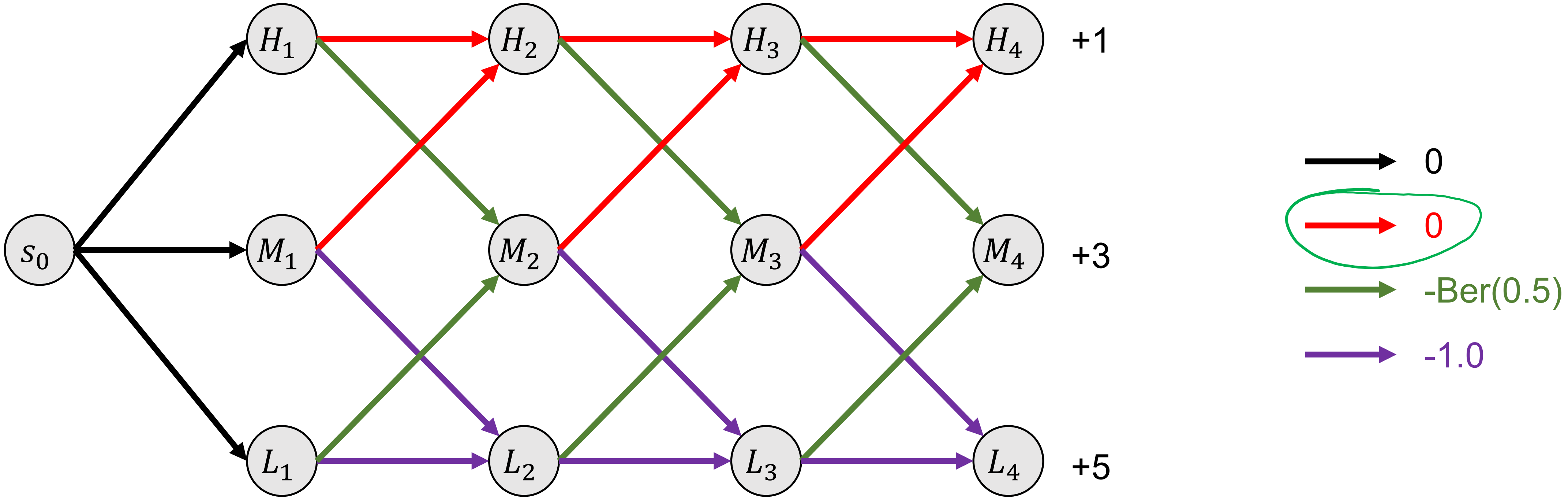
Stochastic Worlds

Now, suppose taking an action does **not** lead to the desired state deterministically.

Instead, with probability 0.8, it goes to the state as specified in the figure;
with probability 0.1 each, it goes to the other two states.



$\pi^*(S)$ for all states



$\boxed{V^\star(H_4)} = 1$ $\quad Q^\star(H_3, R) =$ $\quad V^\star(H_3) =$

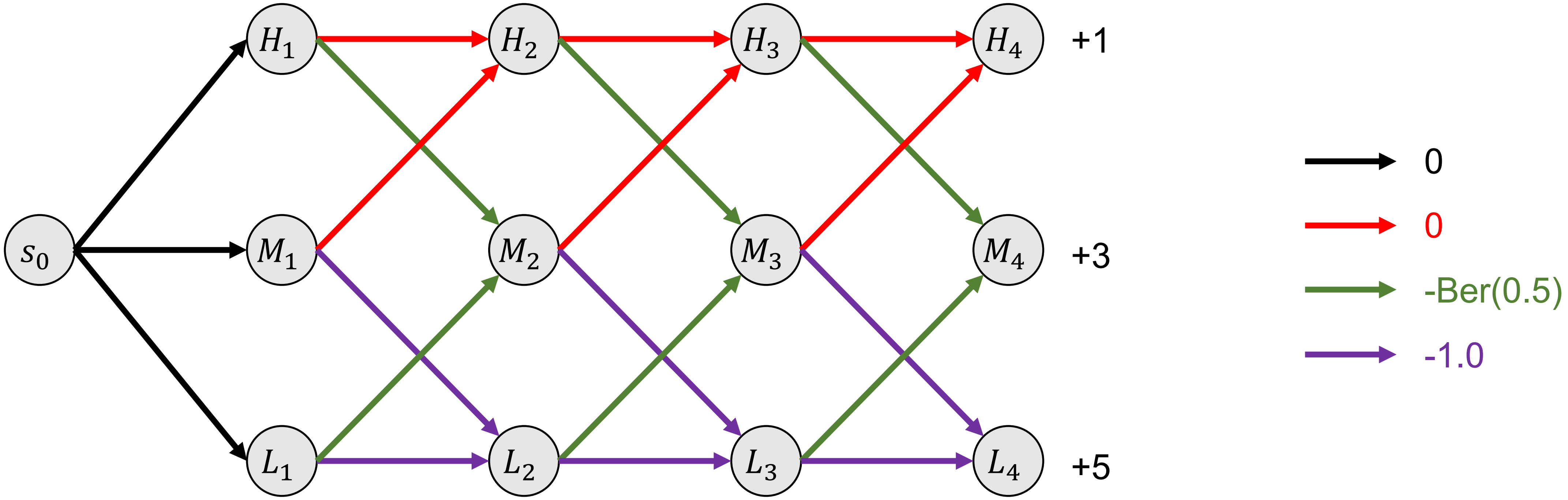
$Q^\star(H_3, G) =$

$V^\star(M_4) = 3$ $\quad Q^\star(M_3, R) =$ $\quad V^\star(M_3) =$

$Q^\star(M_3, P) =$

$V^\star(L_4) = 5$ $\quad Q^\star(L_3, G) =$ $\quad V^\star(L_3) =$

$Q^\star(L_3, P) =$



$V^\star(H_3) =$ $Q^\star(H_2, R) =$ $V^\star(H_2) =$

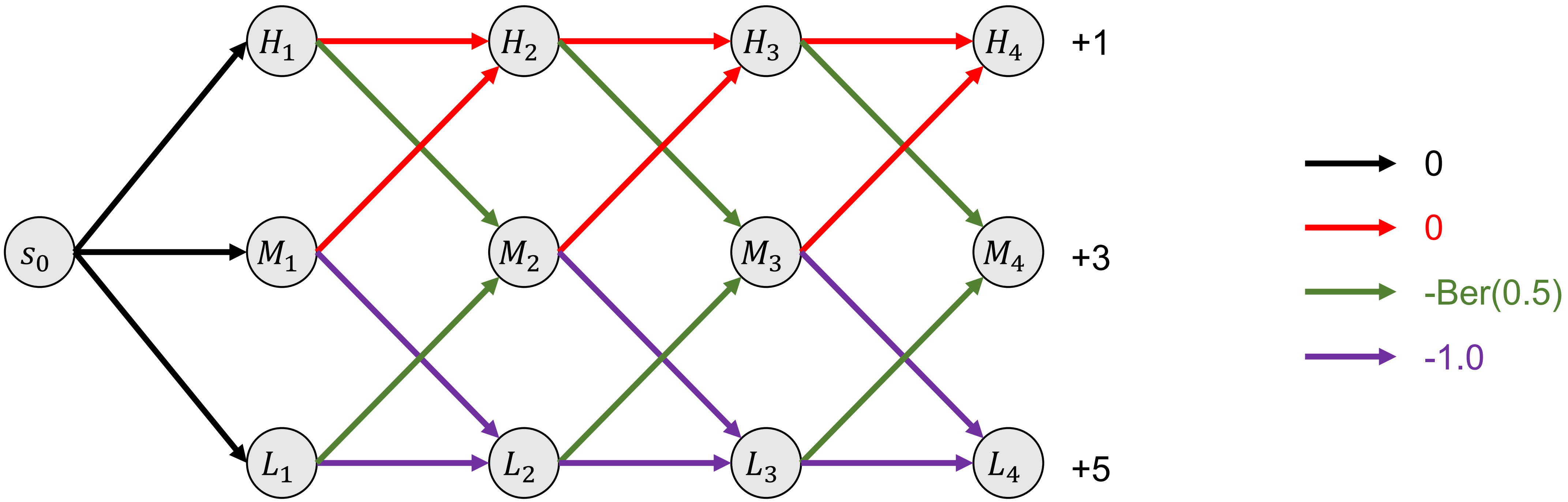
$Q^\star(H_2, G) =$

$V^\star(M_3) =$ $Q^\star(M_2, R) =$ $V^\star(M_2) =$

$Q^\star(M_2, P) =$

$V^\star(L_3) =$ $Q^\star(L_2, G) =$ $V^\star(L_2) =$

$Q^\star(L_2, P) =$



Relation between $Q^\star, V^\star, \pi^\star$:

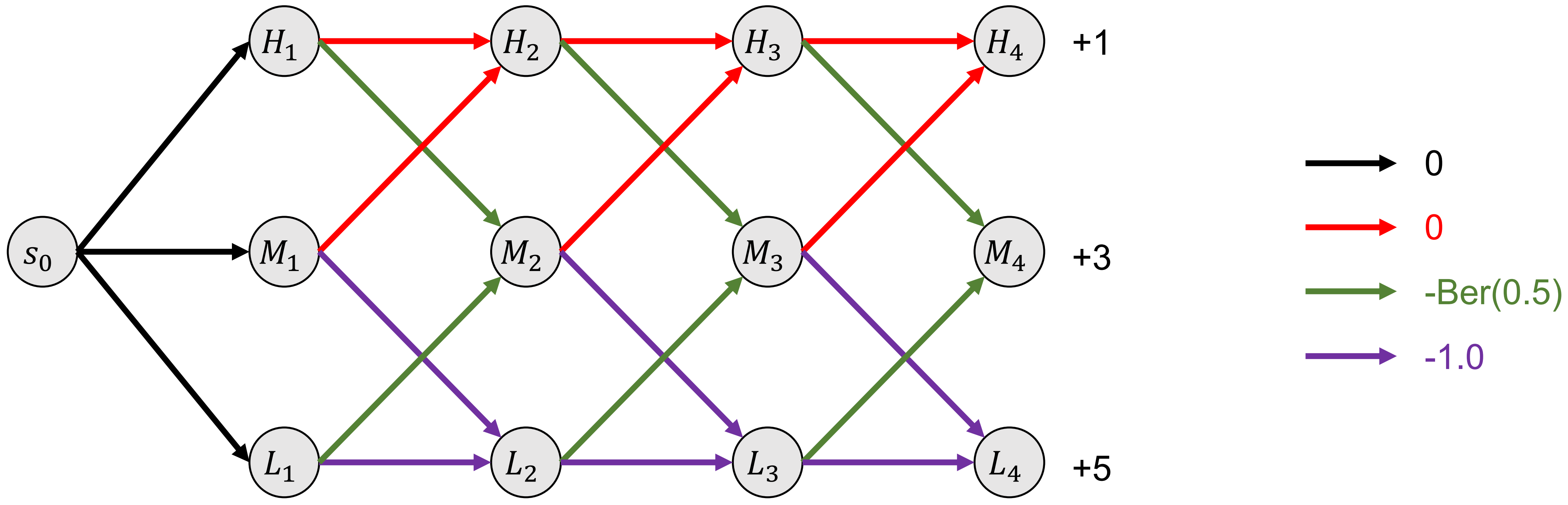
$$Q^\star(s,a) = R(s,a) + \sum_{s'} P(s'|s,a)\, V^\star(s')$$

$$V^\star(s) = \max_a Q^\star(s,a)$$

$$\pi^\star(s) = \operatorname*{argmax}_a Q^\star(s,a)$$

Transition probability $P(s'|s,a)$:

The probability of going to state s' if we take action a on state s



Value Iteration:

```
Repeat until $Q, V$ no longer changes:
        $Q(s,a) \leftarrow R(s,a) + \sum_{s'} P(s'|s,a)\,V(s')$        for all $(s,a)$
        $V(s) = \max_a Q(s,a)$        for all $s$
$\pi^\star(s) = \operatorname{argmax}_a Q(s,a)$
```

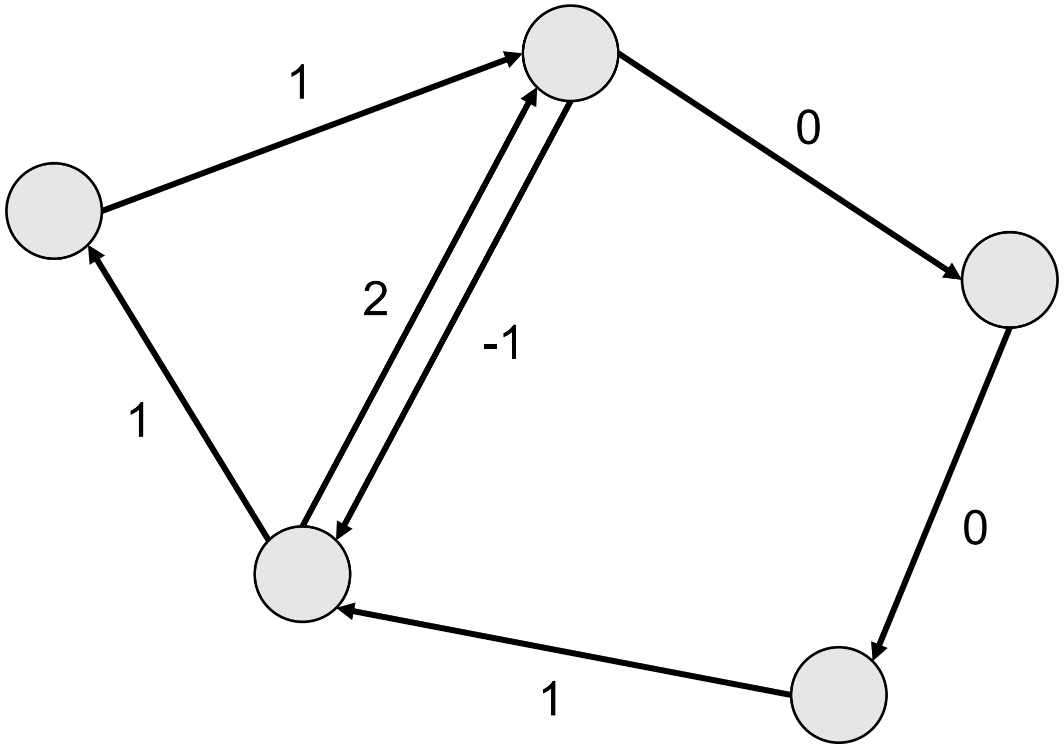
Infinite / Long-Horizon Case

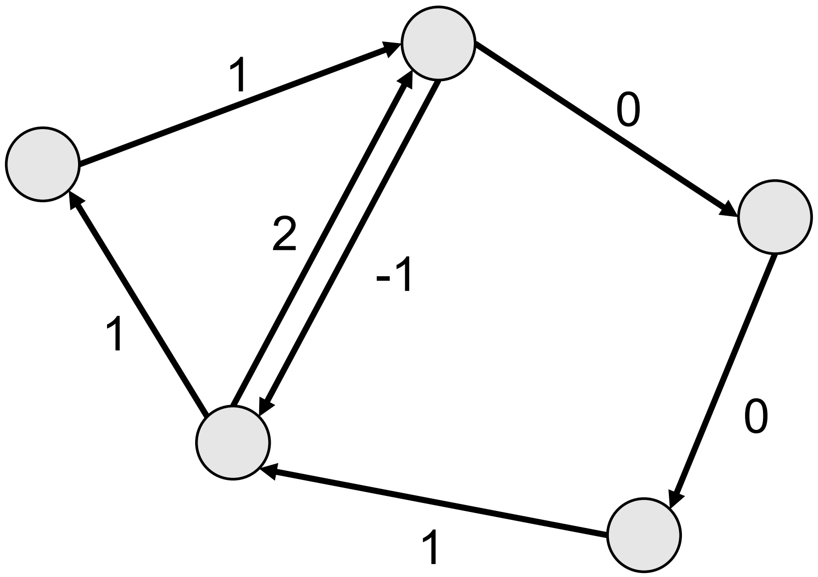
What if there is no “terminal state”?

- Tasks with terminal states
  - Tic-Tac-Toe
  - Tetris
  - Robot doing housework

- Tasks without terminal states (or tasks with very **long horizon** or **loops**)
  - Driving
  - Inventory control

How to gain most reward **in the long run**?   (or, in 1000 rounds)





Repeat until Q, V no longer changes? (Won’t terminate)

$$Q(s,a) \leftarrow R(s,a) + \sum_{s'} P(s'|s,a)\,V(s') \qquad \text{for all } (s,a)$$

$$V(s) = \max_a Q(s,a) \qquad \text{for all } s$$

$$\pi^\star(s) = \operatorname*{argmax}_a Q(s,a)$$

Discounting

$$Q^\star(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^\star(s')$$

$$V^\star(s) = \max_a Q^\star(s,a)$$

$$\pi^\star(s) = \operatorname*{argmax}_a Q^\star(s,a)$$

To calculate $Q^\star, V^\star, \pi^\star$ --- see later slide for a more detailed version

Repeat until Q, V **becomes stable:**

$$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V(s') \quad \text{for all } (s,a)$$

$$V(s) = \max_a Q(s,a) \quad \text{for all } s$$

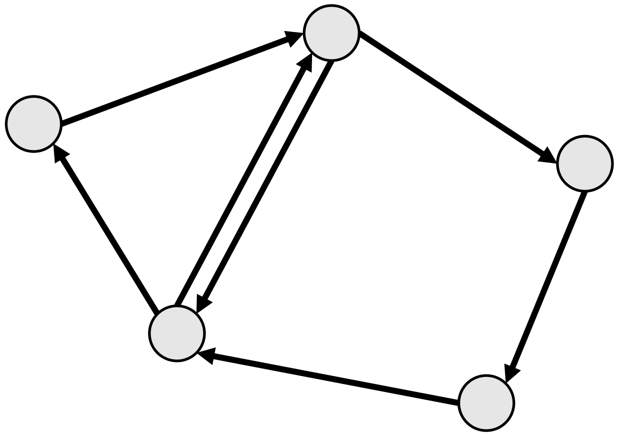
$$\pi^\star(s) = \operatorname*{argmax}_a Q(s,a)$$

Discounting: Equivalent View

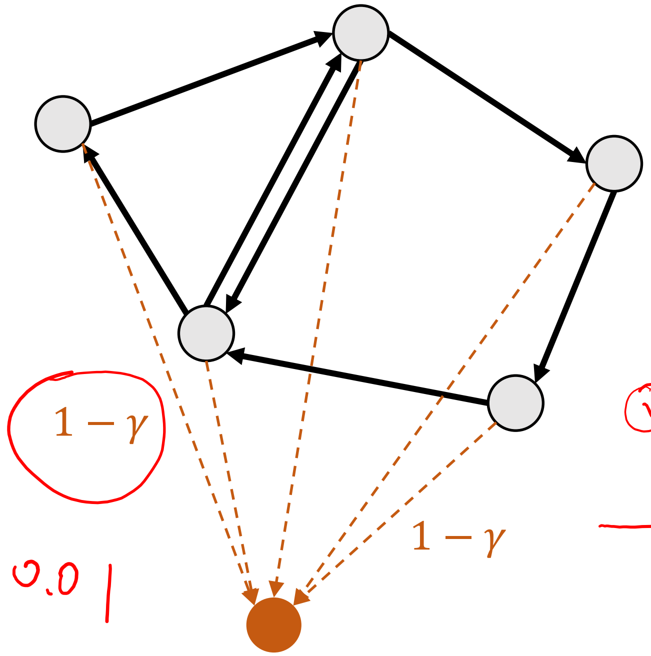
p up

$1-p$ down

$E[\text{time before the first facing up}] = \frac{1}{p}$



Original Transition



Consider the expected total reward in this modified graph.

$Q^*(s,a) \le O\left(\frac{1}{1-\gamma}\right) \approx 100$

Modified Transition: on any state, taking any action,

$$\text{Next state} = \begin{cases} \text{terminal state} & \text{with probability } 1-\gamma \\ \text{following } P(s'|s,a) & \text{with probability } \gamma \end{cases}$$

Discounting

V^*, Q^*

Effective horizon length (the expected length before going to the dummy terminal state) = $\frac{1}{1-\gamma}$

- Goal of discounting: Make $V^\star, Q^\star$ finite even when there are positive loops
  - γ should be strictly smaller than 1
- The modification should still approximate the learner's goal well enough
  - γ should be sufficiently close to 1

Usually, γ is set to some value like 0.99, but it depends on the application

Discounting

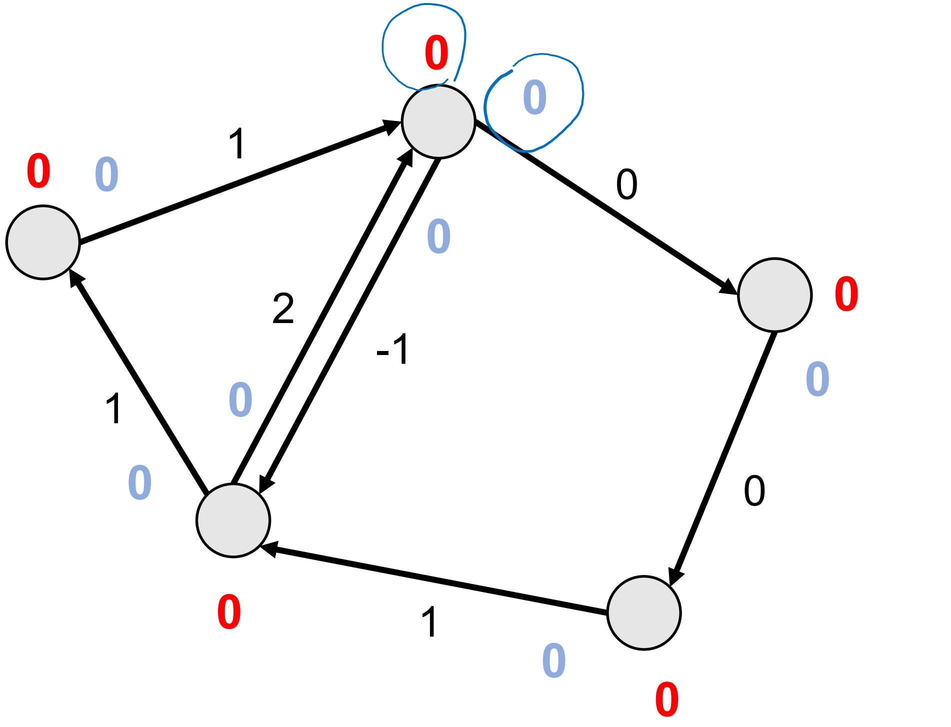
Value Iteration with discounting

$\frac{1}{1-\gamma}$

```
Initialize $Q_0(s,a) \leftarrow 0,\ V_0(s) \leftarrow 0$   for all $(s,a)$
For $i = 1, 2, \ldots$
    $Q_i(s,a) \leftarrow R(s,a) + \gamma V_{i-1}(\text{next\_state}(s,a))$   for all $(s,a)$
    $V_i(s) = \max_a Q_i(s,a)$   for all $s$
    If $\underline{|Q_i(s,a) - Q_{i-1}(s,a)| \leq \underset{0{,}0001}{\epsilon}}$ for all $(s,a)$:   break
$\pi^\star(s) = \operatorname{argmax}_a Q_{\text{final}}(s,a)$
```

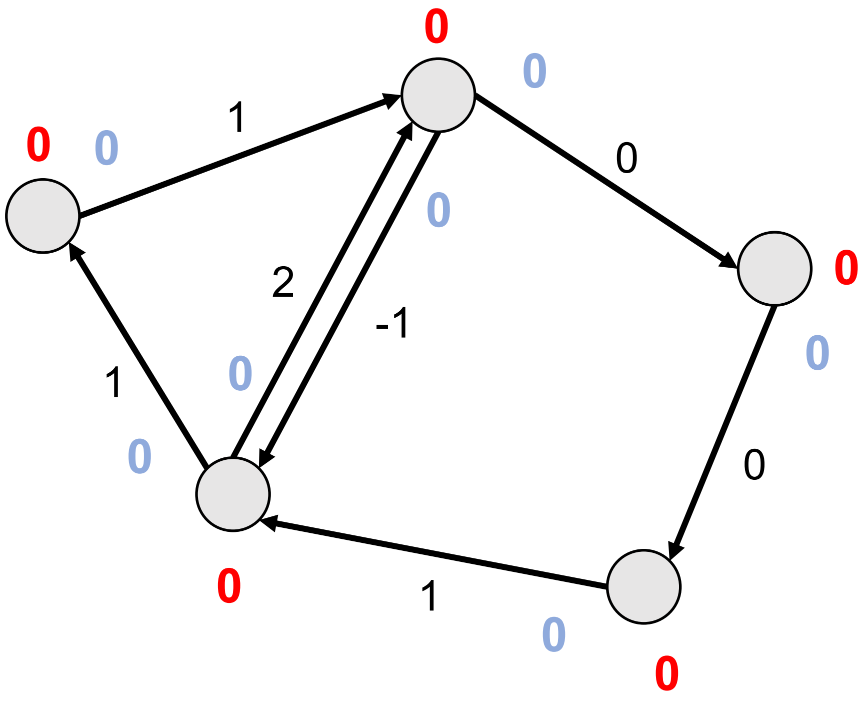
$Q_0(s,a) = 0$

$V_0(s) = 0$



$\gamma = 0.95$

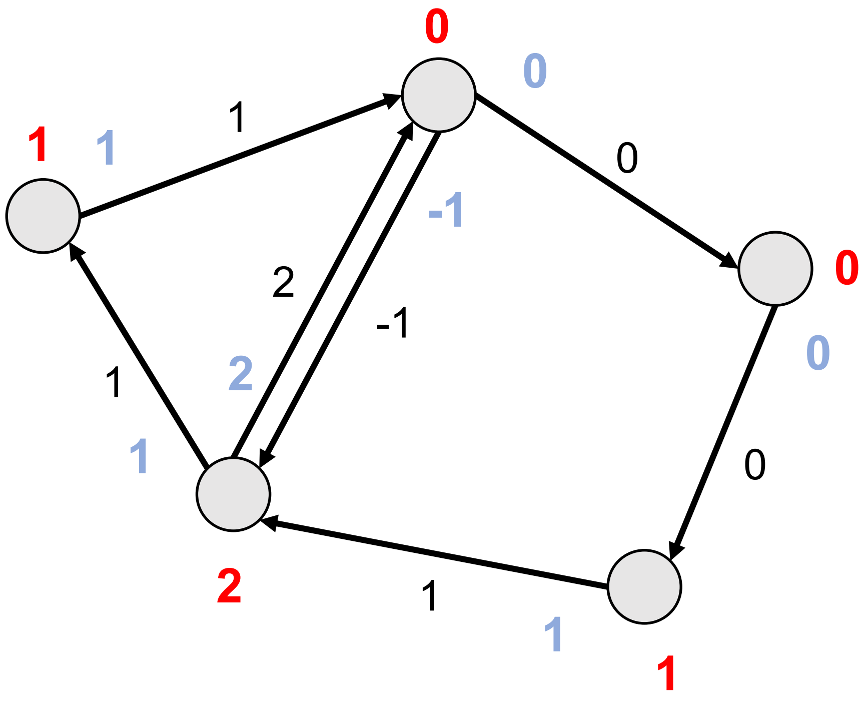
$\frac{1}{1-\gamma} = 20$



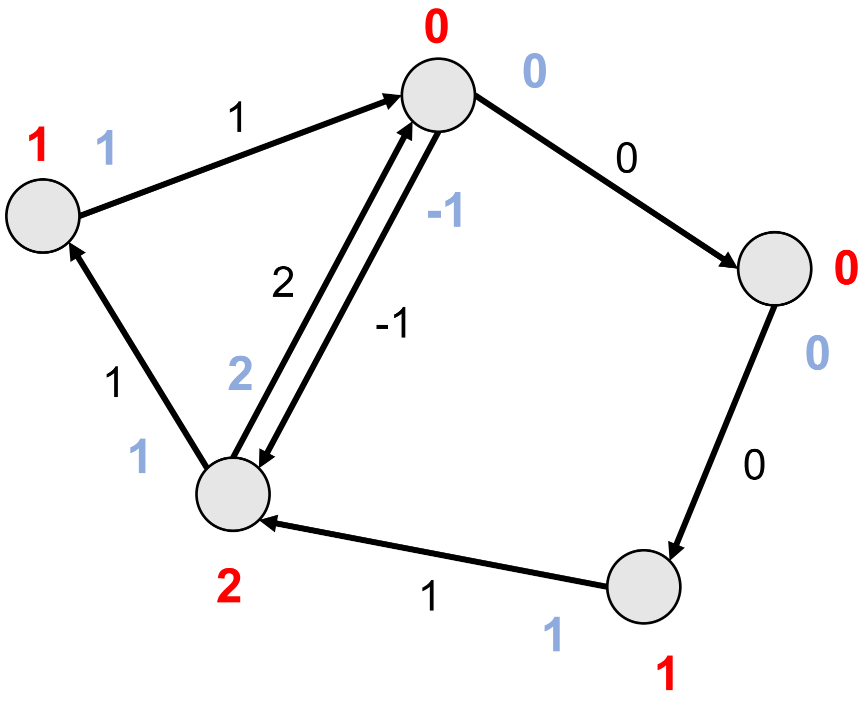
$\gamma = 1.0$

$$\underline{Q_i(s,a)} \leftarrow \underline{R(s,a)} + \gamma \underbrace{\sum_{s'} P(s'|s,a)\, V_{i-1}(s')}$$

$$\boxed{V_i(s) \leftarrow \max_a Q_i(s,a)}$$



$\gamma = 0.95$

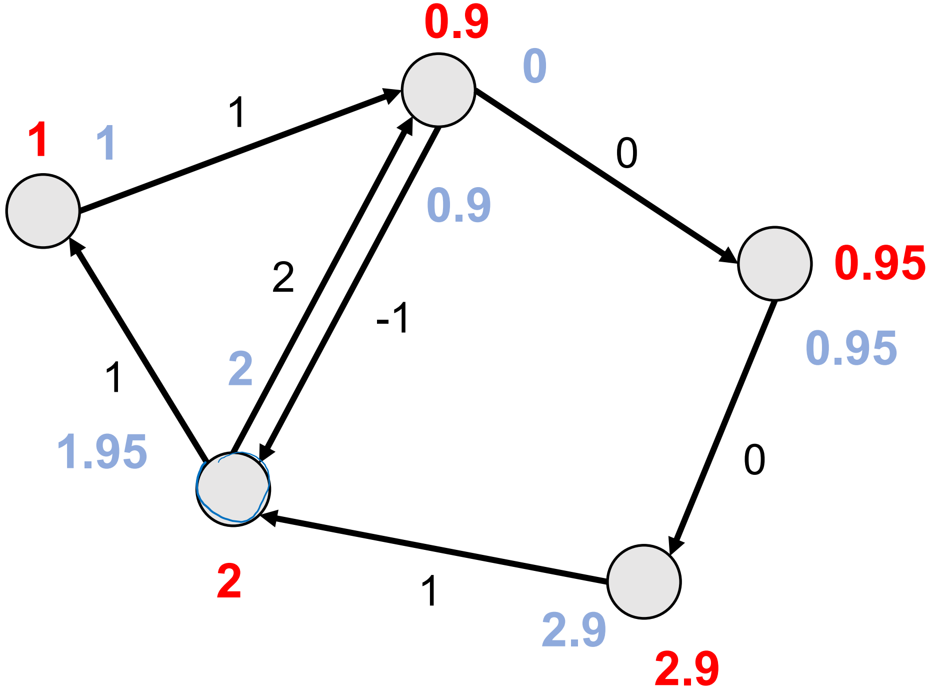


$\gamma = 1.0$

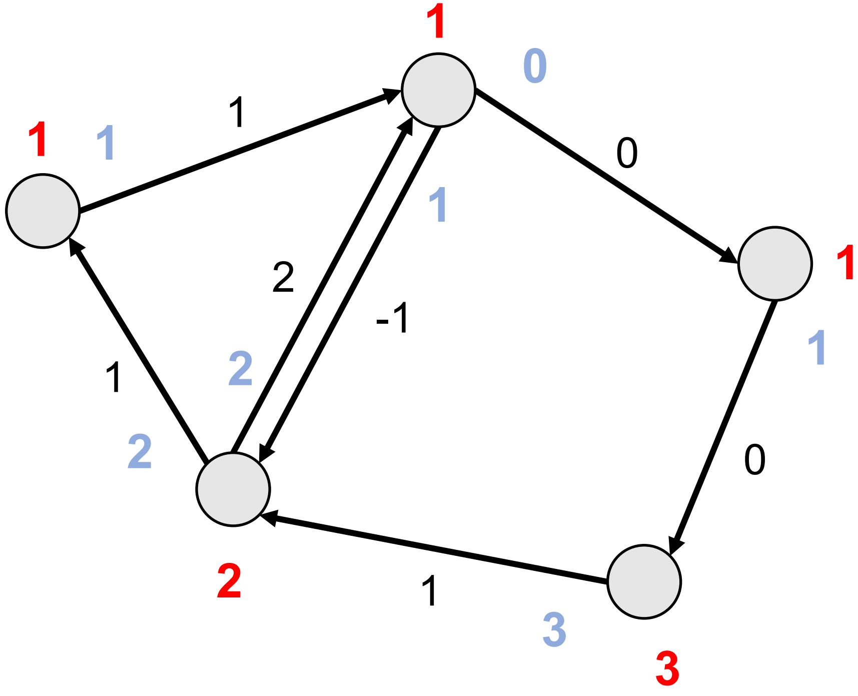
Iteration $i = 1$

$$Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$V_i(s) \leftarrow \max_a Q_i(s,a)$$



$\gamma = 0.95$

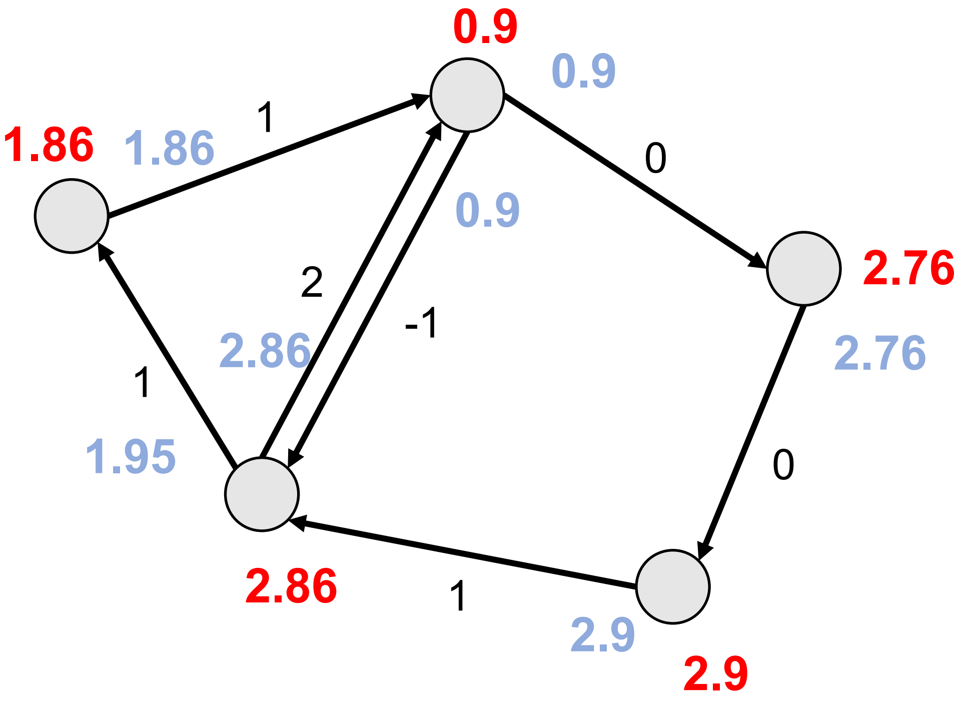


$\gamma = 1.0$

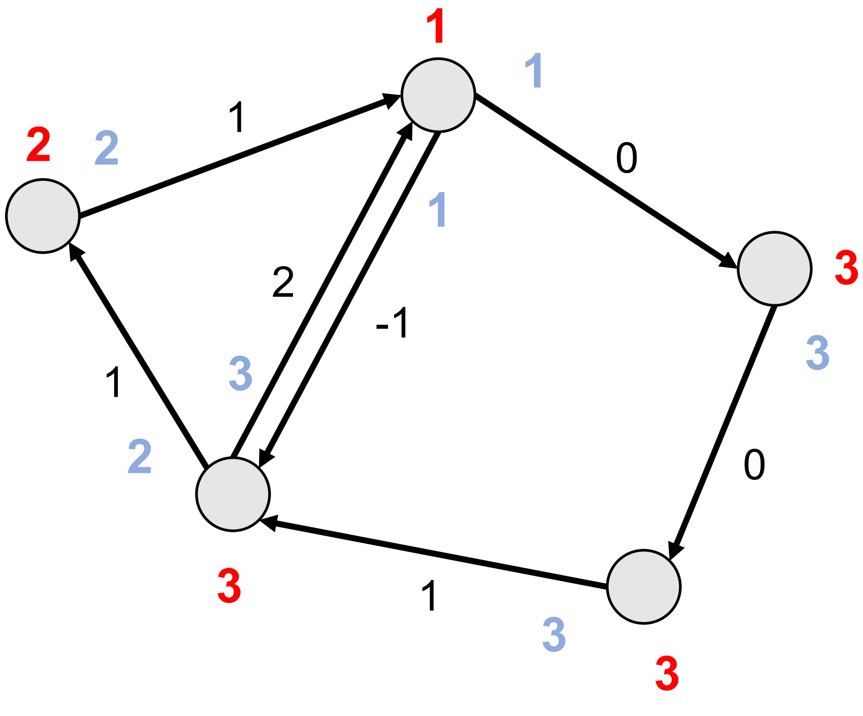
Iteration $i = 2$

$$Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$V_i(s) \leftarrow \max_a Q_i(s,a)$$



$\gamma = 0.95$

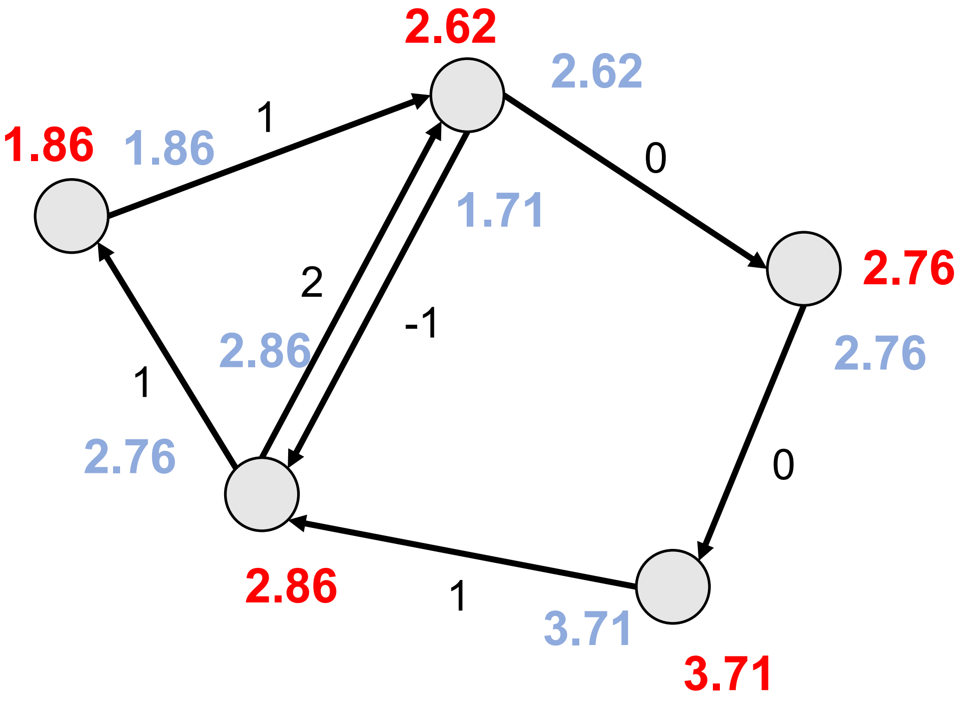


$\gamma = 1.0$

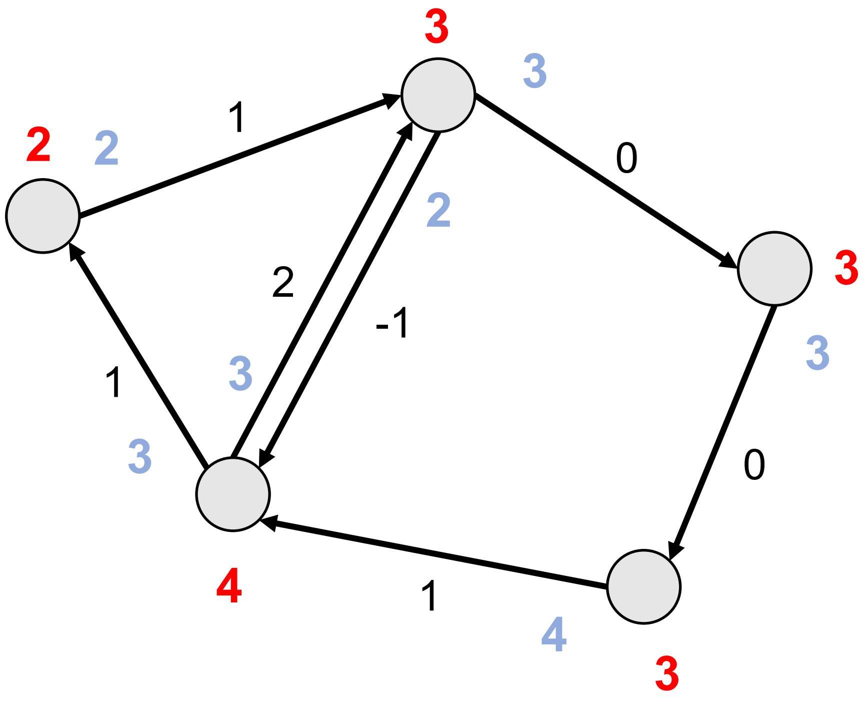
Iteration $i = 3$

$$Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$V_i(s) \leftarrow \max_a Q_i(s,a)$$



$\gamma = 0.95$

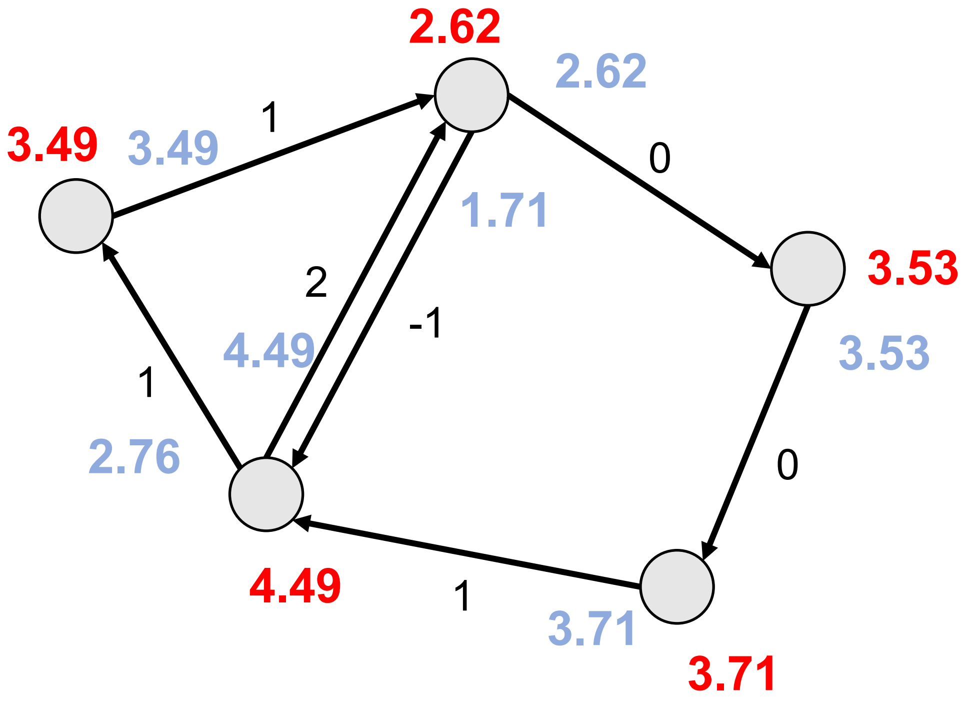


$\gamma = 1.0$

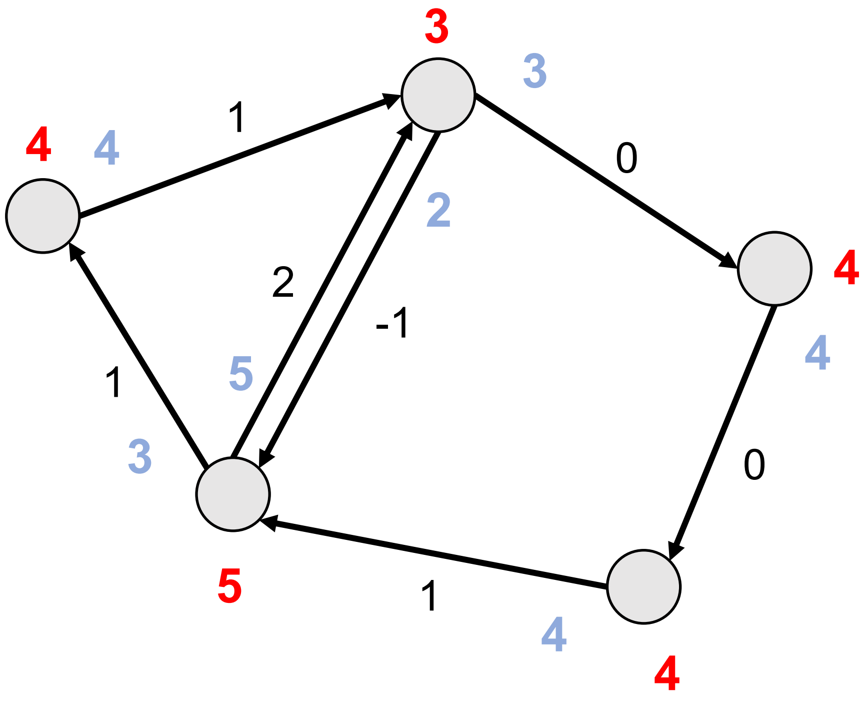
Iteration $i = 4$

$$Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$V_i(s) \leftarrow \max_a Q_i(s,a)$$



$\gamma = 0.95$

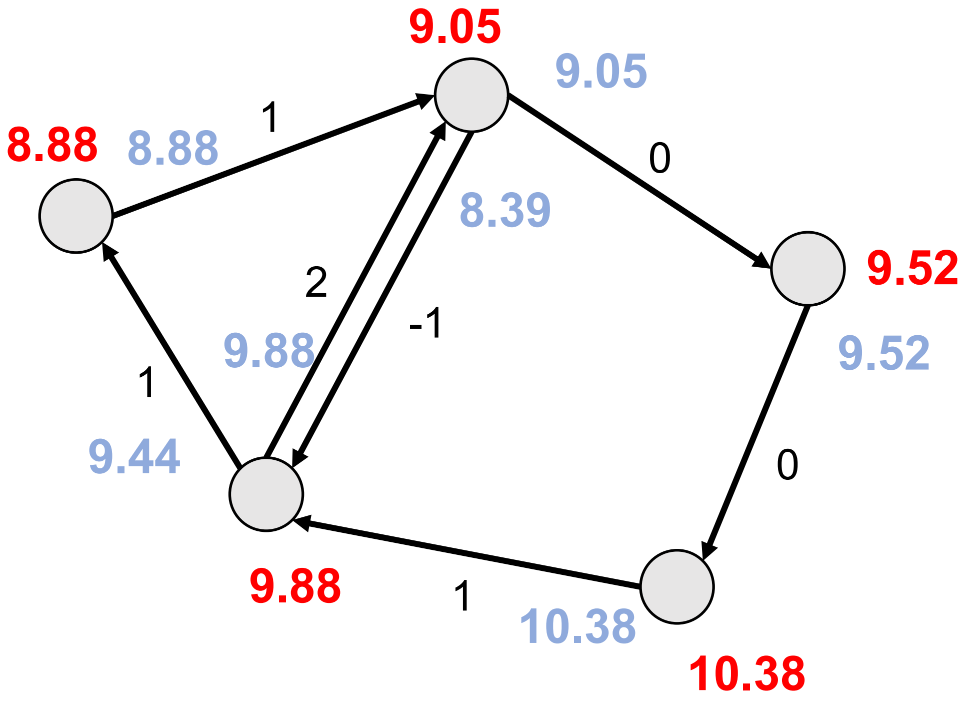


$\gamma = 1.0$

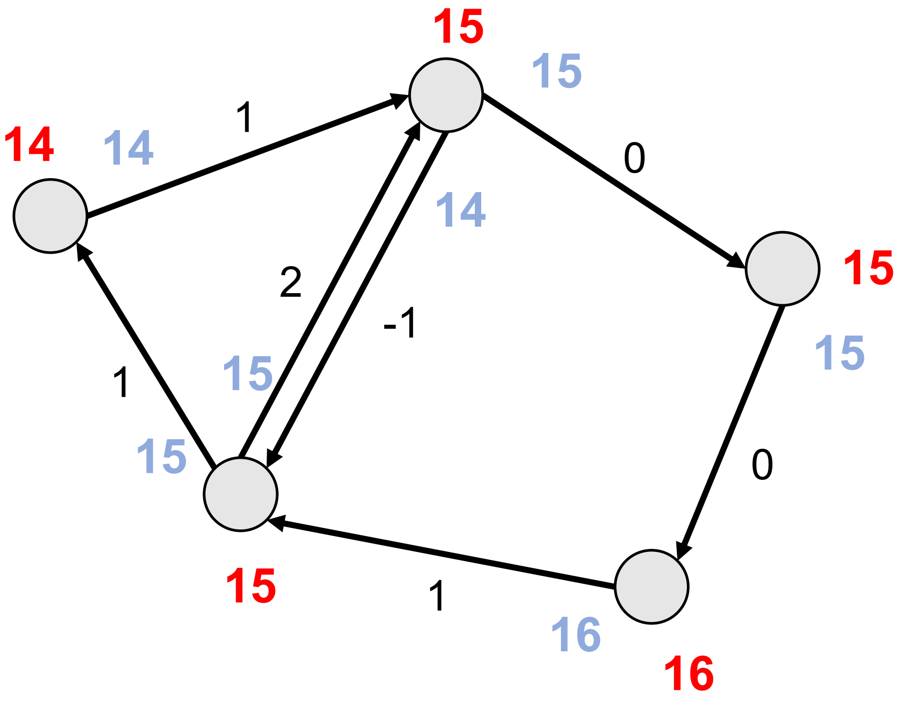
Iteration $i = 5$

$$Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$V_i(s) \leftarrow \max_a Q_i(s,a)$$



$\gamma = 0.95$

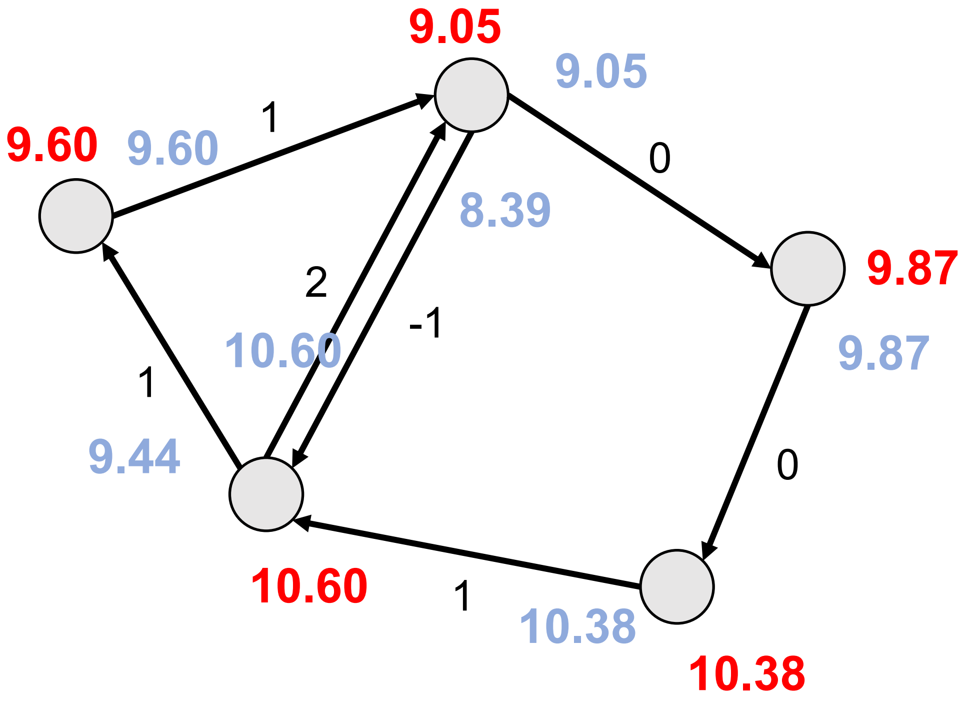


$\gamma = 1.0$

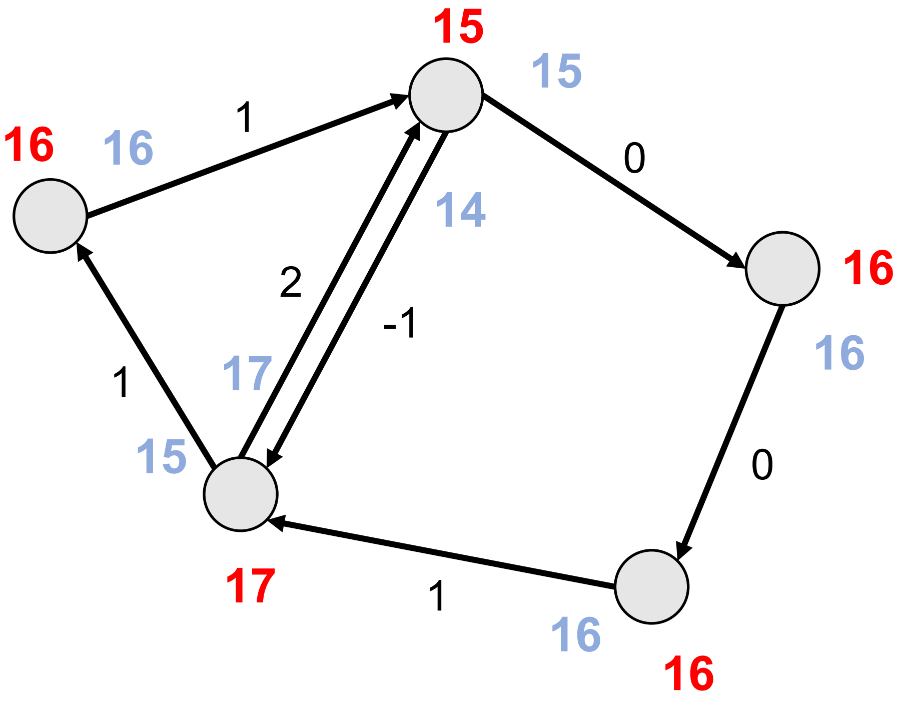
Iteration $i = 20$

$$Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$V_i(s) \leftarrow \max_a Q_i(s,a)$$



$\gamma = 0.95$

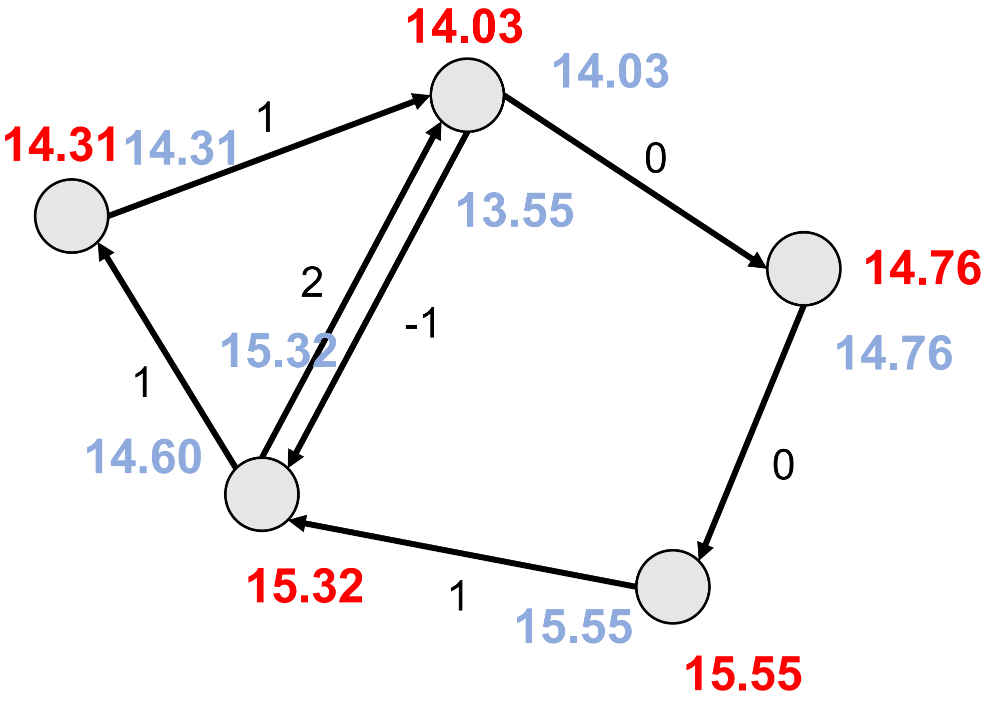


$\gamma = 1.0$

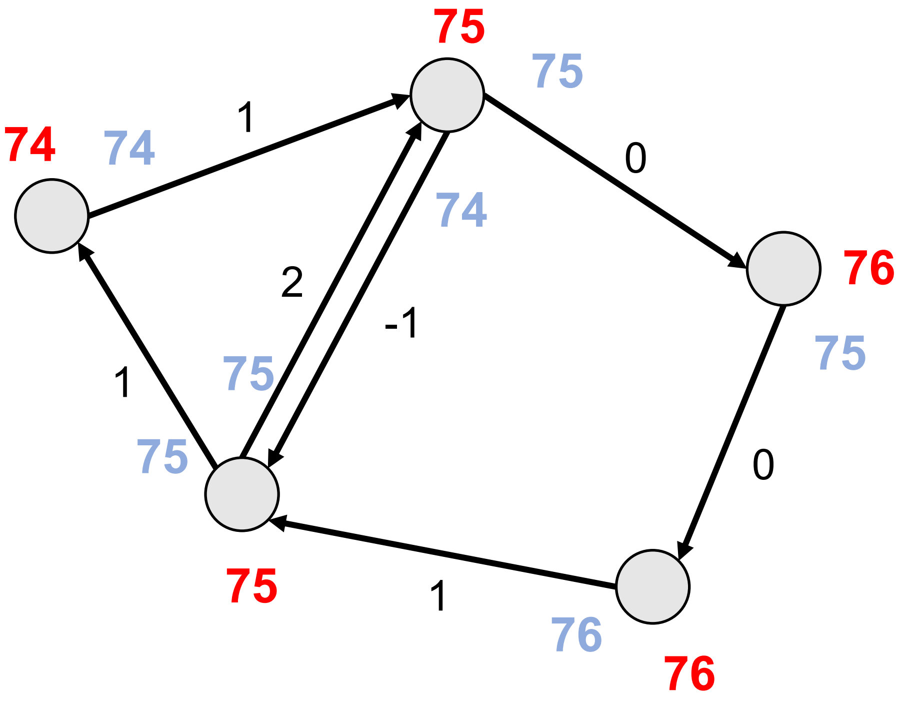
Iteration $i = 21$

$$\boldsymbol{Q_i(s,a)} \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$\boldsymbol{V_i(s)} \leftarrow \max_a Q_i(s,a)$$



$\gamma = 0.95$



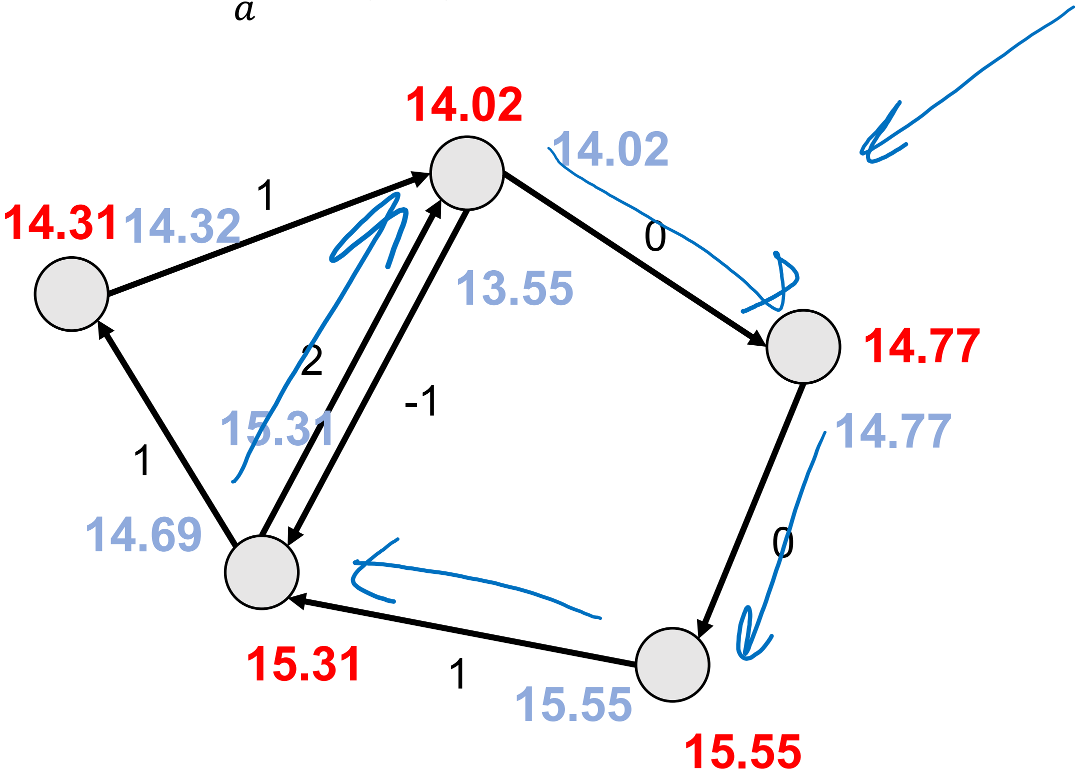
$\gamma = 1.0$

Iteration $i = 100$

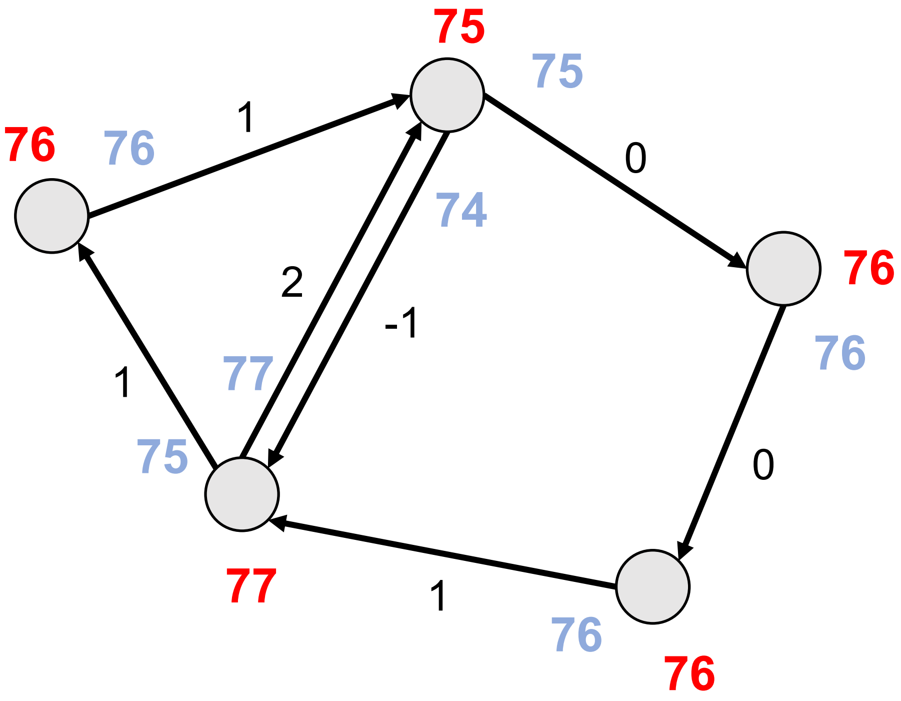
$$Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$$

$$V_i(s) \leftarrow \max_a Q_i(s,a)$$

$$\boxed{\frac{1}{1-\gamma} = 20}$$



$\gamma = 0.95$



$\gamma = 1.0$

Iteration $i = 101$

Markov Decision Process (MDP)

Elements:

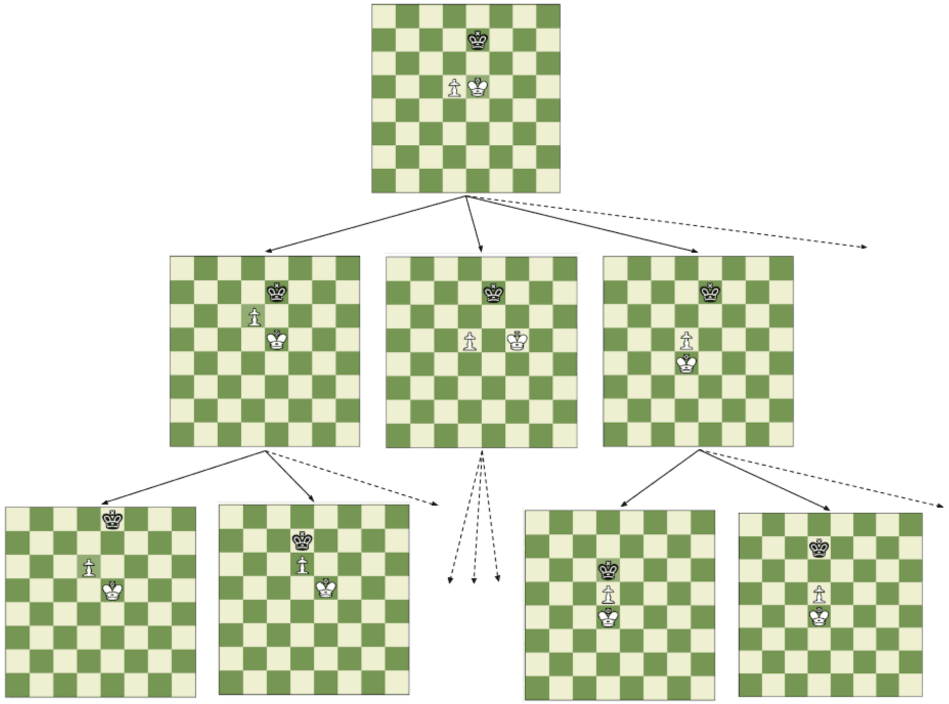
- Reward function $R(s, a)$
- Transition probability $P(s'|s, a)$
- Set of terminal states $\mathcal{T}$ (may be empty)

Parameter:

- Discount factor $\gamma \in [0,1]$: set of 1 only if you know that every episode will be short; otherwise, set it to something close to 1
  - Smaller (e.g., 0.9): easier to converge, but only find policy that optimize shorter-term objective
  - Larger (e.g., 0.995): take longer to converge, but find policy that optimizes longer-term objective

Markov Decision Process (MDP)

- Modern RL formulates sequential decision making as MDPs

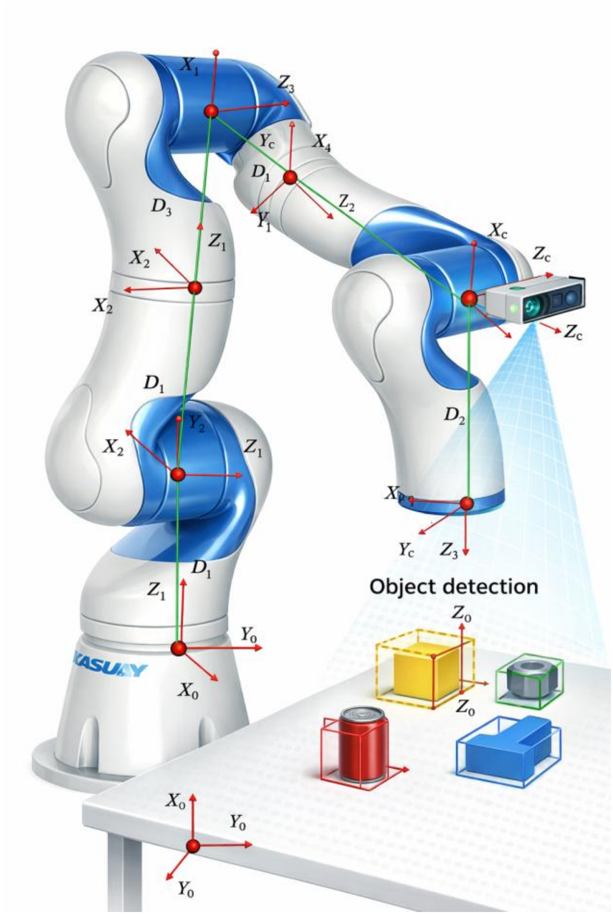


As in contextual bandits, as there are too many possible states/contexts, the learner must be able to handle **unseen state/contexts.**

For example, use a neural network $Q_\theta(s, a)$ to approximate $Q^\star(s, a)$

Markov Decision Process (MDP)

- Modern RL formulates sequential decision making as MDPs



States:

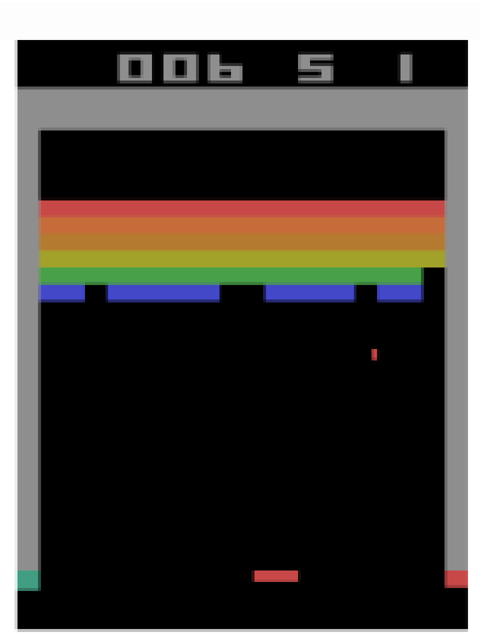
- Summarize the past actions (can ignore irrelevant details)
- Provide sufficient information for the next decision

Contexts in contextual bandits vs. **States** in MDPs:

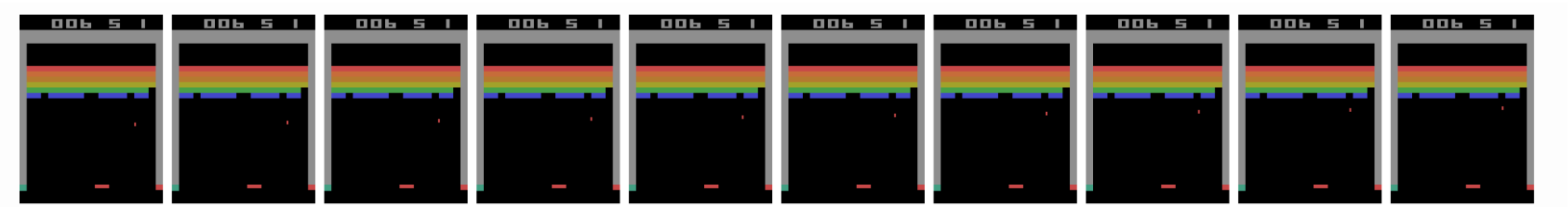
- Same role: the information needed to make (next) decision
- States are further affected by the learner's own past actions

Markov Decision Process (MDP)

- Modern RL formulates sequential decision making as MDPs



Single frame: may not be sufficient for next decision



Stacking frames: so we know the velocity and direction of the ball

May use low resolution / black-white images
(no need to include less relevant information)

$Q^\star/V^\star$ versus Q^π/V^π

What are $Q^\star / V^\star$ (Review)

$V^\star(s) :=$ maximum expected total reward starting from state s

$Q^\star(s,a) :=$ maximum expected total reward starting from state s and taking action a **for one step**, and then following the optimal strategy

Relation:

$$Q^\star(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^\star(s')$$

$$V^\star(s) = \max_a Q^\star(s,a)$$

How to find / approximate them:

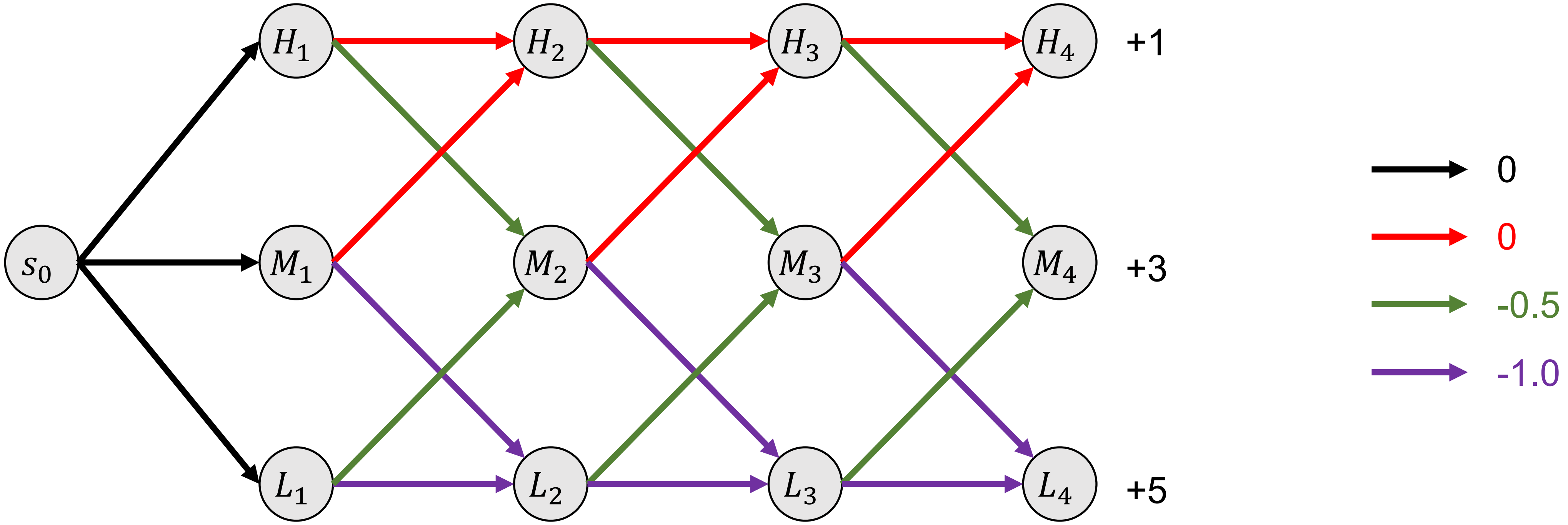
```
Initialize  $Q_0(s,a) \leftarrow 0,\ V_0(s) \leftarrow 0$   for all $(s,a)$
For $i = 1, 2, \ldots$
        $Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$   for all $(s,a)$
        $V_i(s) = \max_a Q_i(s,a)$   for all $s$
        If $|Q_i(s,a) - Q_{i-1}(s,a)| \le \epsilon$ for all $(s,a)$:   break
```

$\boldsymbol{Q^{\pi}/V^{\pi}}$

Fix a policy π

$V^{\pi}(s) :=$ expected total reward starting from state s and following policy π

$Q^{\pi}(s, a) :=$ expected total reward starting from state s and taking action a for one step, and then following policy π



Fix a policy π:

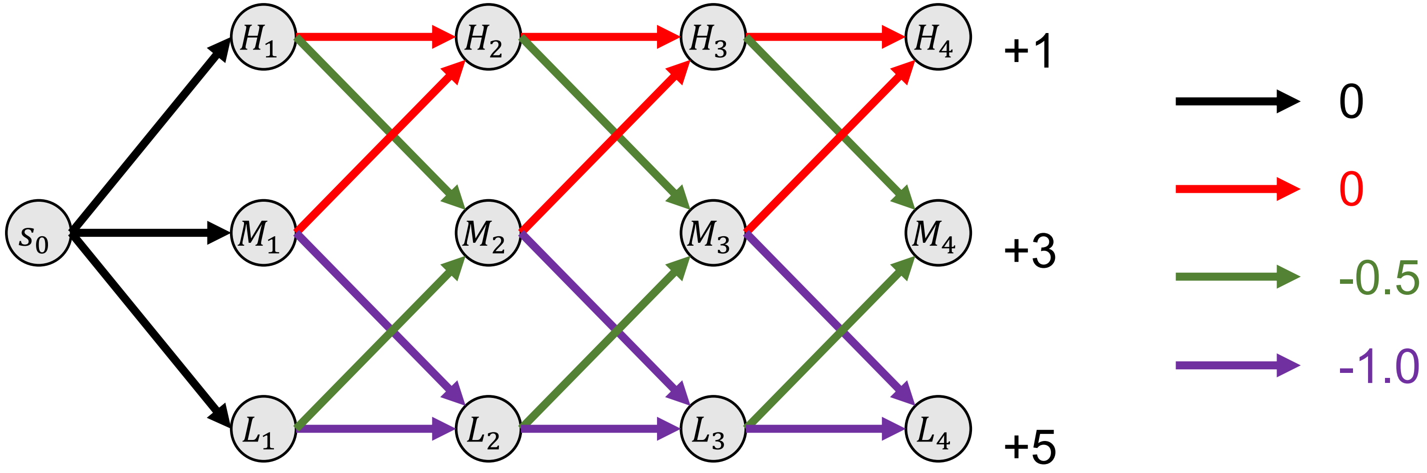
$$\pi(R \mid H_i) = \frac{1}{2}\,, \qquad \pi(G|H_i) = \frac{1}{2}$$

$$\pi(R \mid M_i) = 0\,, \qquad \pi(P|M_i) = 1$$

$$\pi(G \mid L_i) = \frac{1}{2}, \qquad \pi(P|L_i) = \frac{1}{2}$$

$$\pi(\cdot \mid s_0) = \text{Uniform}$$

What is the expected total reward of policy π?



$$\pi(R \mid H_i) = \frac{1}{2}\,, \qquad \pi(G|H_i) = \frac{1}{2}$$

$$\pi(R \mid M_i) = 0\,, \qquad \pi(P|M_i) = 1$$

$$\pi(G \mid L_i) = \frac{1}{2}, \qquad \pi(P|L_i) = \frac{1}{2}$$

$$\pi(\cdot \mid s_0) = \text{ Uniform}$$

Fixed policy π

$V^\pi(H_4) =$

$V^\pi(M_4) =$

$V^\pi(L_4) =$

$Q^\pi(H_3, R) =$

$Q^\pi(H_3, G) =$

$Q^\pi(M_3, R) =$

$Q^\pi(M_3, P) =$

$Q^\pi(L_3, G) =$

$Q^\pi(L_3, P) =$

$V^\pi(H_3) =$

$V^\pi(M_3) =$

$V^\pi(L_3) =$

$Q^\pi(H_2, R) =$

$Q^\pi(H_2, G) =$

$Q^\pi(M_2, R) =$

$Q^\pi(M_2, P) =$

$Q^\pi(L_2, G) =$

$Q^\pi(L_2, P) =$

Q^{π}/V^{π}

Fix a policy π

$V^{\pi}(s) \coloneqq$ expected total reward starting from state s and following policy π

$Q^{\pi}(s,a) \coloneqq$ expected total reward starting from state s and taking action a for one step, and then following policy π

Relation:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^{\pi}(s')$$

$$V^{\pi}(s) = \sum_{a} \pi(a|s)\, Q^{\pi}(s,a)$$

How to find / approximate them:

```
Initialize $Q_0(s,a) \leftarrow 0,\ V_0(s) \leftarrow 0$ for all $(s,a)$
For $i = 1, 2, \ldots$
    $Q_i(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_{i-1}(s')$ for all $(s,a)$
    $V_i(s) = \sum_{a} \pi(a|s)\, Q_i(s,a)$ for all $s$
    If $|Q_i(s,a) - Q_{i-1}(s,a)| \leq \epsilon$ for all $(s,a)$: break
```

Why do we introduce Q^{π}/V^{π} in addition to $Q^{\star}/V^{\star}$

- In some scenarios, we care about evaluating the performance of a given policy (rather than finding the optimal policy)
- As will see in later lectures, there are two RL algorithm design principles:
  - Approximating $Q^{\star}/V^{\star}$ with neural networks
  - Approximating Q^{π}/V^{π} with neural networks, where π is the current policy of the learner
- They have different pros and cons

Where are we now?

We have introduced a framework (Markov Decision Process) to model sequential decision making problems.

Comparison with contextual bandits:

- Contextual bandits: find a policy to maximize $R(x, a)$
  - Given R, the optimal policy is simple to derive: choose $\pi^\star(x) = \mathrm{argmax}_a\, R(x, a)$
- MDP: Find a policy to maximize $R(s_1, a_1) + R(s_2, a_2) + \cdots + R(s_\tau, a_\tau)$
  - Given R and P, the optimal policy can be derived or approximated by dynamic programming / value iteration

Next:

- Consider bandit feedback (exploration) and neural network (generalization) again, designing practical RL algorithm with sequential decisions