Reinforcement Learning Training 2025

Bellman's Equation

> Foundation to solving MDP and RL problems.

Recall (1)

- Markov decision process has transition probabilities

$$\mathbf{Pr}\Big[S_{t+1} = s', R_{t+1} = r \Big| S_t = s, A_t = a\Big]$$

which transitions the agen to state S_{t+1} and a reward of R_{t+1}

- Cumulative reward at time t

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

Recall (2)

- A value function is an expected cumulative reward

$$v_\pi(s) = \mathbf{E}_\pi[G_t | S_t = s]$$

- A action-value value function is an expected cumulative reward from taking action a

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$$

> Note that v and q depend on the policy π.

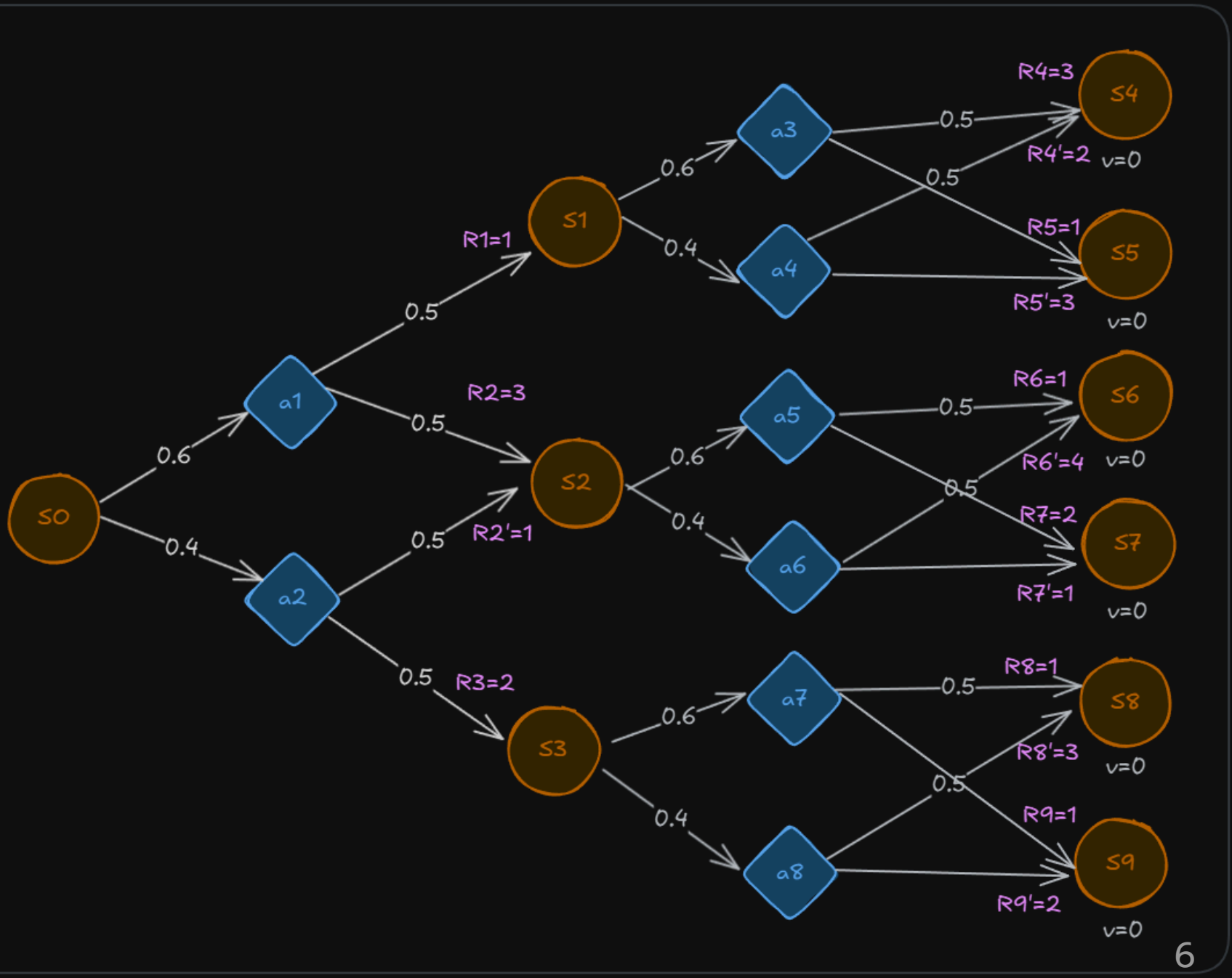
Bellman Equation

- Allows relationships among v and q.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma\, v_\pi(s')\right]$$

$$q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q(s', a')\right]$$

Example


s0
a1
a2
s1
s2
s3
a3
a4
a5
a6
a7
a8
s4
s5
s6
s7
s8
s9
0.6
0.4
0.5
R1=1
R2=3
R2'=1
R3=2
R4=3
R4'=2
R5=1
R5'=3
R6=1
R6'=4
R7=2
R7'=1
R8=1
R8'=3
R9=1
R9'=2
v=0

Find $v(s_1)$

$$V(s_1) = 0.6\left[0.5 \times (R_4 + \gamma V(s_4)) + 0.5 \times (R_5 + \gamma V(s_5))\right]$$
$$+ 0.4\left[0.5 \times (R_4' + \gamma V(s_4)) + 0.5 \times (R_5' + \gamma V(s_5))\right]$$

$$V(s_1) = 0.6\left[0.5(3) + 0.5(1)\right] + 0.4\left[0.5(2) + 0.5(3)\right]$$

$$= 2.2$$

Find $v(s_2)$ and $v(s_3)$

$$V(s_2) = 0.6\,[0.5(1) + 0.5(2)] + 0.4\,[0.5(4) + 0.5(1)] = 1.9$$

$$V(s_3) = 0.6\,[0.5(1) + 0.5(1)] + 0.4\,[0.5(3) + 0.5(2)] = 1.6$$

Find $v(s_0)$

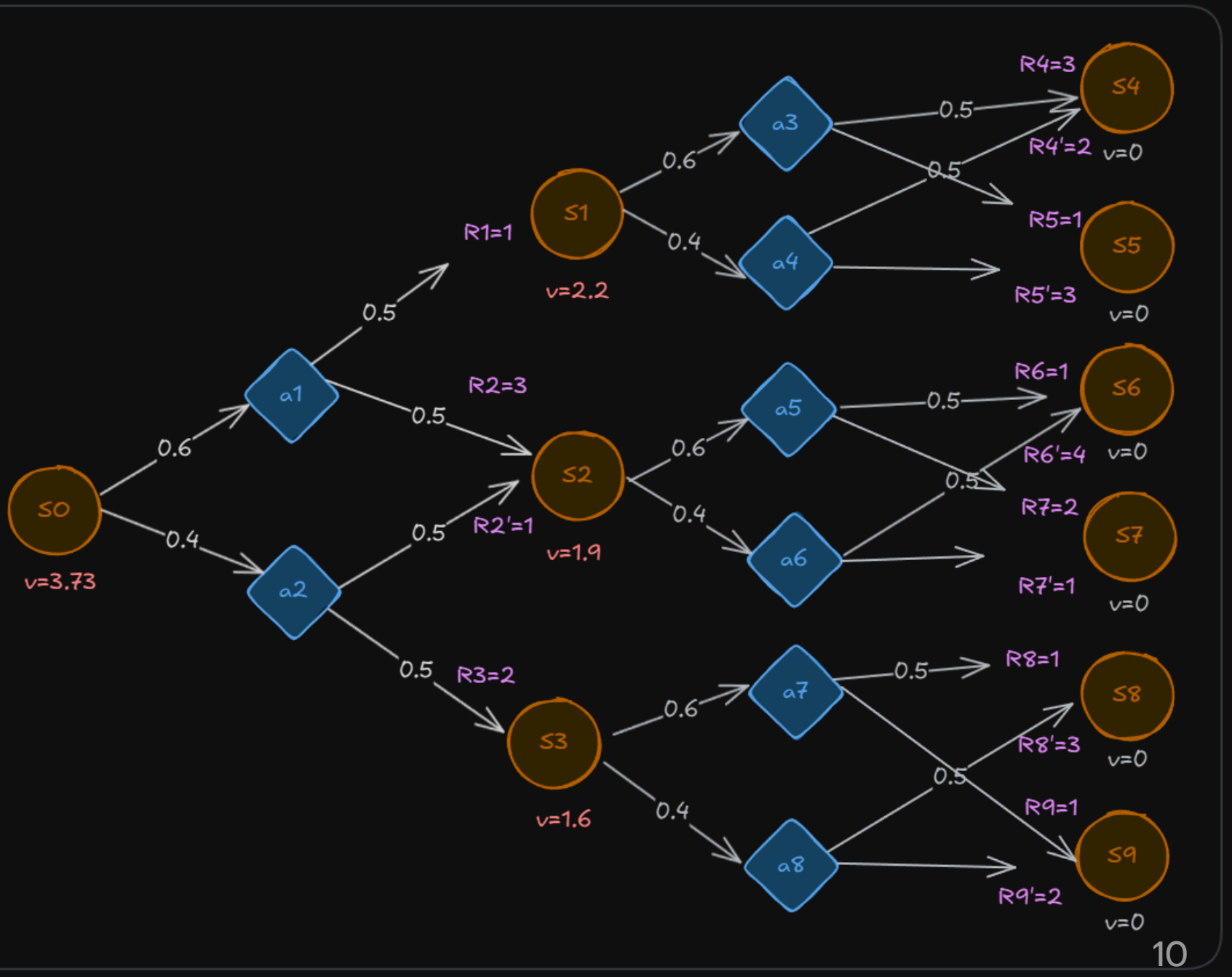
$$V(S_0) = 0.6\left[0.5(1+2.2) + 0.5(3+1.9)\right] + 0.4\left[0.5(1+1.9) + 0.5(2+1.6)\right]$$

$$= 0.6(4.25) + 0.4(3.25)$$

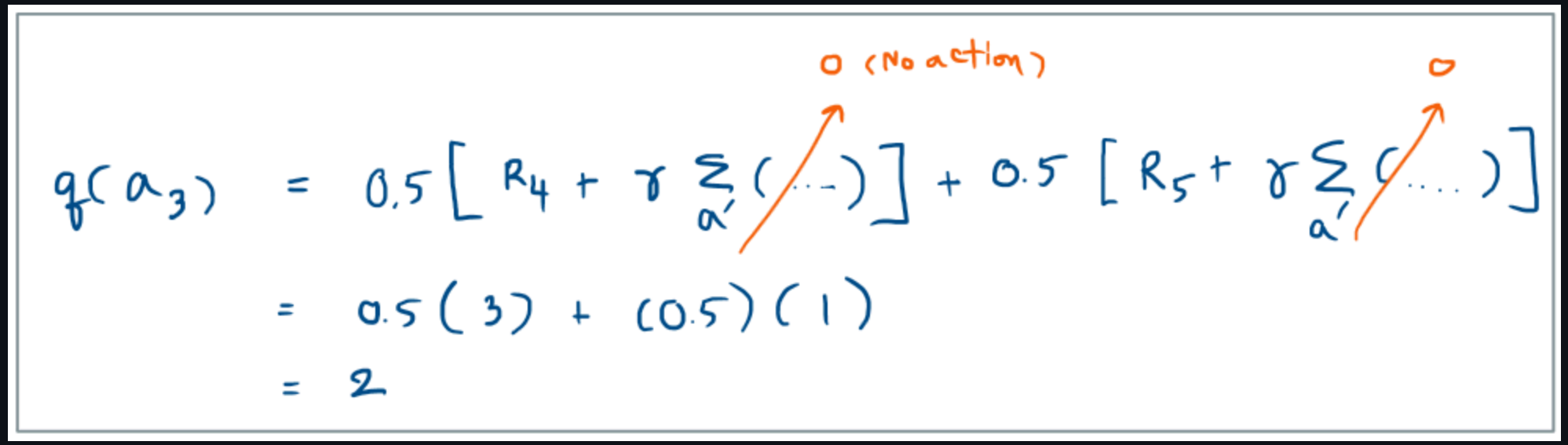
$$= 3.73$$

Result

See values of v



Find $q(a_3)$


0 (No action)
0
$q(a_3) = 0.5\left[R_4 + \gamma \sum_{a'}(\dots)\right] + 0.5\left[R_5 + \gamma \sum_{a'}(\dots)\right]$
$= 0.5(3) + (0.5)(1)$
$= 2$

Find $q(a_4) - q(a_8)$

$q(a_4) = 0.5(2) + 0.5(3) = 2.5$

$q(a_5) = 0.5(1) + 0.5(2) = 1.5$

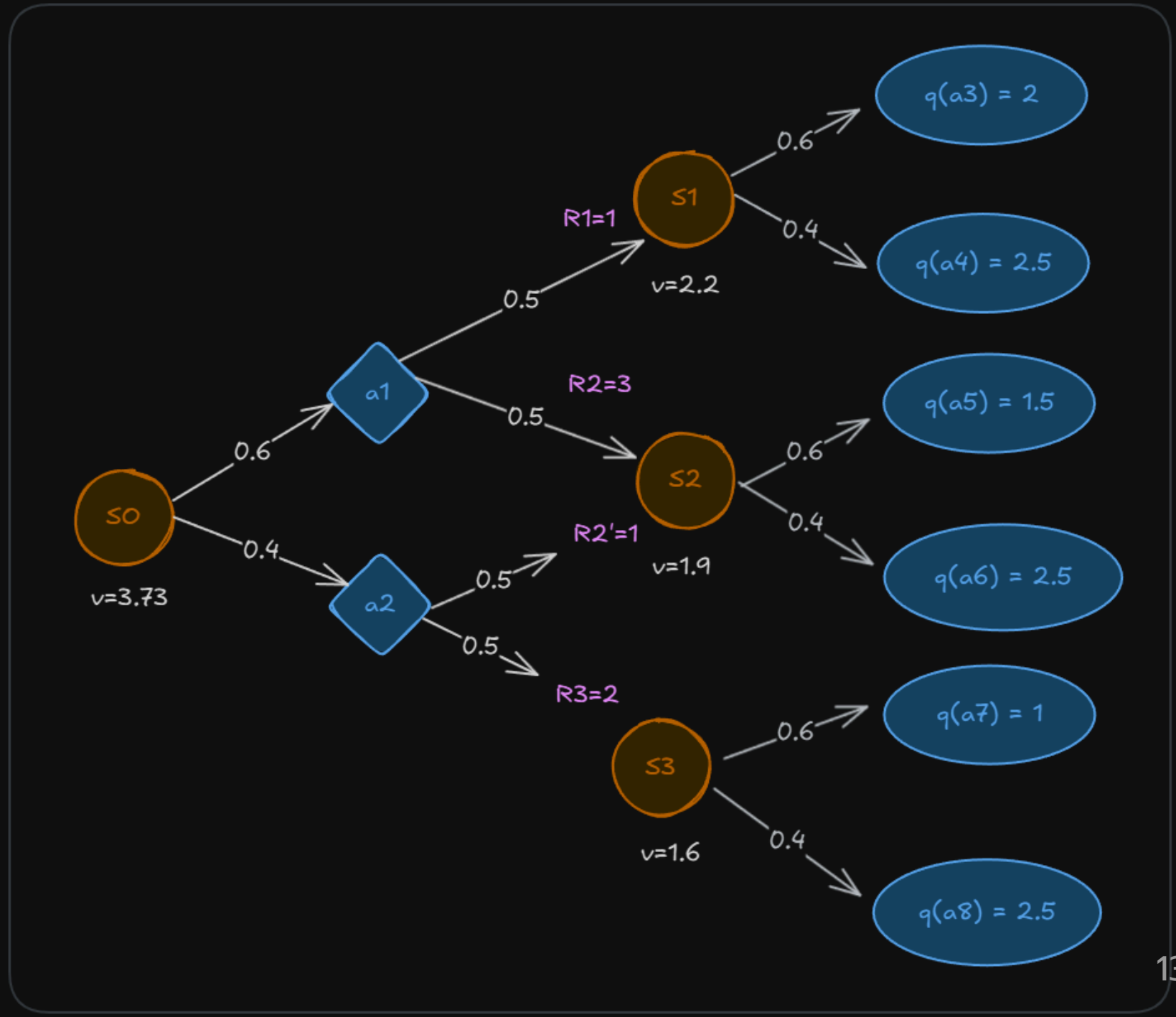
$q(a_6) = 0.5(4) + 0.5(1) = 2.5$

$q(a_7) = 0.5(1) + 0.5(1) = 1$

$q(a_8) = 0.5(3) + 0.5(2) = 2.5$

Result

See q

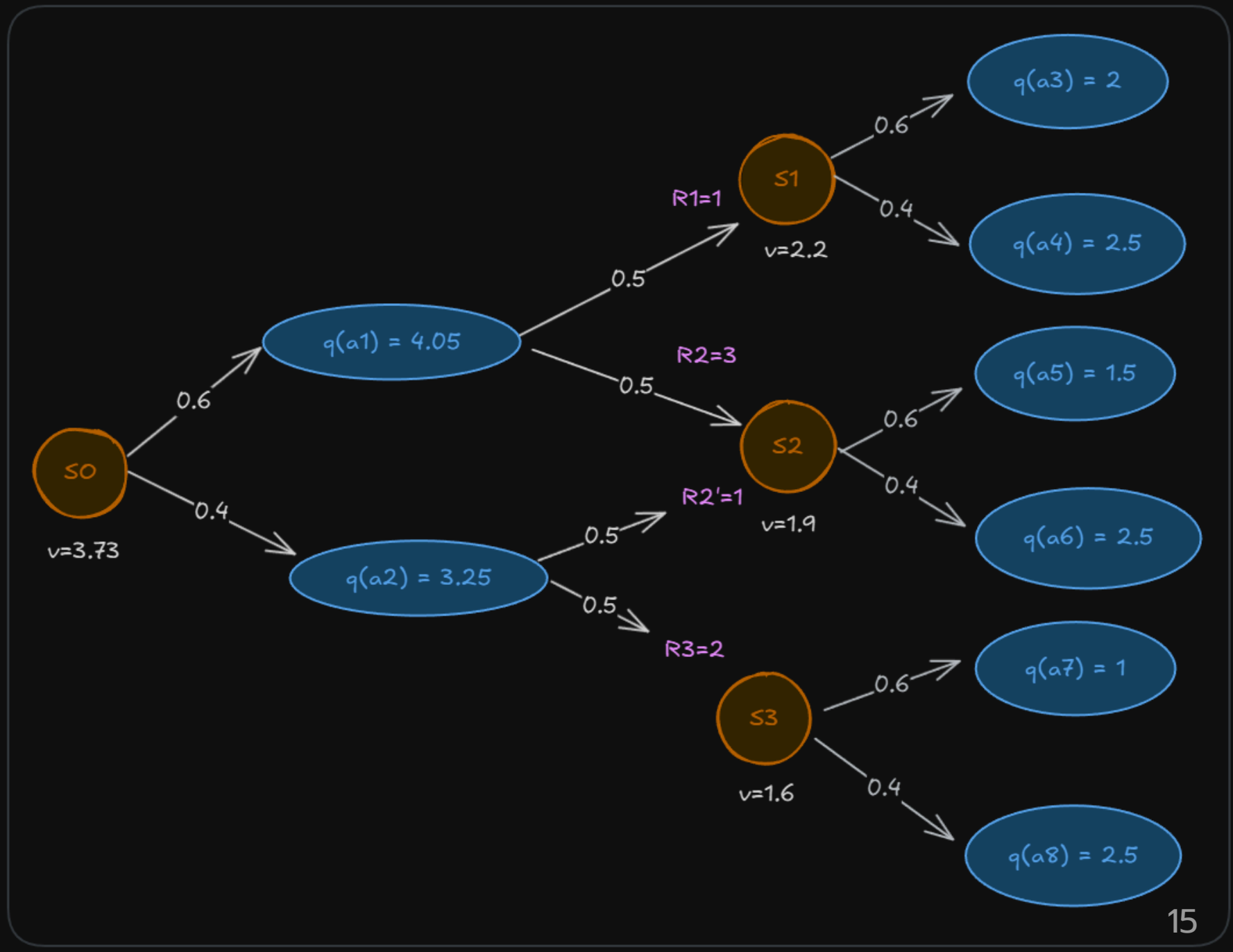

s0
v=3.73
0.6
0.4
a1
a2
0.5
0.5
R1=1
s1
v=2.2
0.6
0.4
q(a3) = 2
q(a4) = 2.5
R2=3
R2'=1
s2
v=1.9
0.5
0.5
0.6
0.4
q(a5) = 1.5
q(a6) = 2.5
R3=2
s3
v=1.6
0.6
0.4
q(a7) = 1
q(a8) = 2.5

$ Find $q(a_1), q(a_2)$

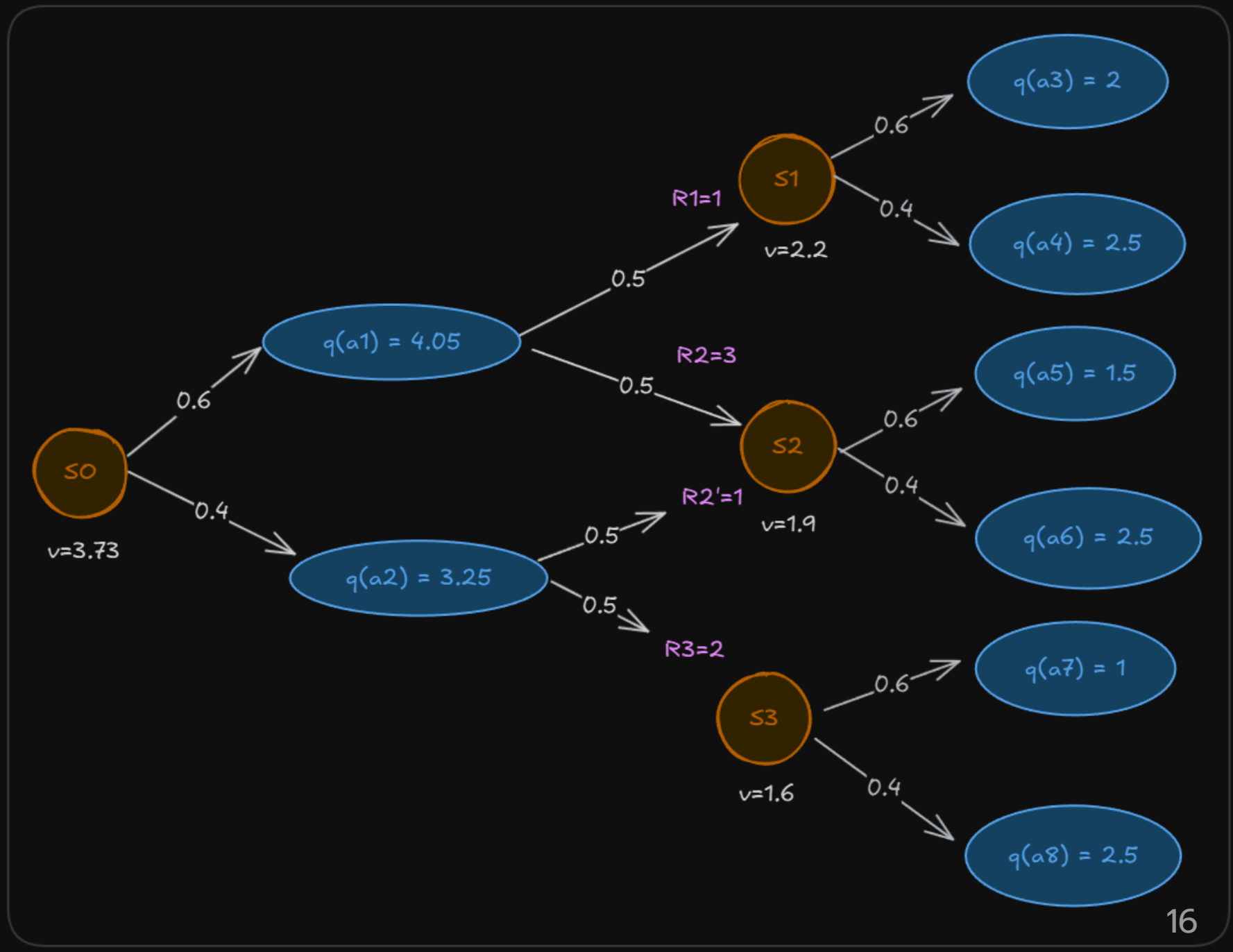
$$q(a_1) = 0.5\left[1 + \cancel{\gamma}^{1}\left(0.6(2) + 0.4(2.5)\right)\right]$$
$$+ 0.5\left[3 + \cancel{\gamma}^{1}\left(0.6(1.5) + 0.4(2.5)\right)\right]$$
$$= 4.05$$

$$q(a_2) = 0.5\left[1 + \gamma\left(0.6(1.5) + 0.4(2.5)\right)\right]$$
$$+ 0.5\left[2 + \gamma\left(0.6(1) + 0.4(2.5)\right)\right]$$
$$= 3.25$$

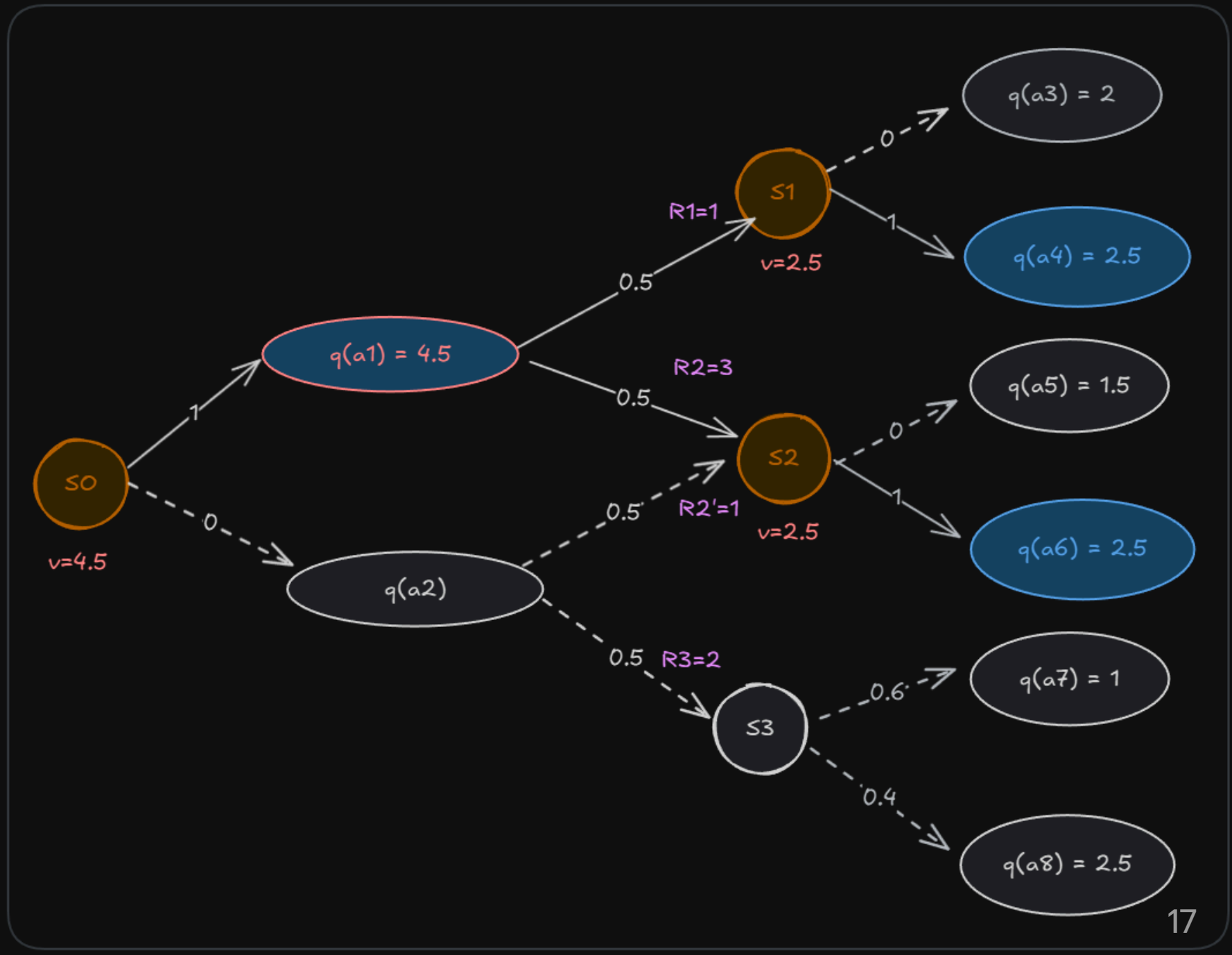
Result


S0
v=3.73
0.6
0.4
q(a1) = 4.05
q(a2) = 3.25
0.5
0.5
R1=1
R2=3
R2'=1
R3=2
S1
v=2.2
S2
v=1.9
S3
v=1.6
0.6
0.4
0.6
0.4
0.6
0.4
q(a3) = 2
q(a4) = 2.5
q(a5) = 1.5
q(a6) = 2.5
q(a7) = 1
q(a8) = 2.5

Can we do better?



Optimal Policy



Optimality Condition

- If agent is following the optimal policy π^* (something we want to find), then the value function will also be optimal.

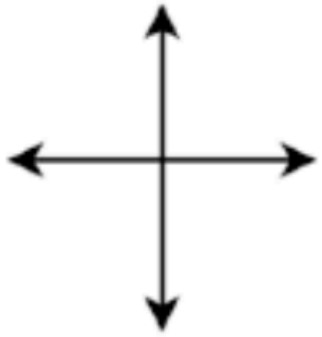
$$v^*(s) = \max_\pi v_\pi(s)$$

- It follows that

$$v^*(s) = \max_a \left\{ \sum_{s',r} p(s', r|s, a)[r + \gamma v^*(s')] \right\}$$

$$q^*(s, a) = \sum_{s',r} p(s', r|s, a) \cdot \left[r + \gamma \max_{a'} q^*(s', a') \right]$$

Gridworld Environment



Actions

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

16 states represented as "0" to "15"
reward = -1 for all transitions

Gridworld

- The top-left and bottom-right positions are terminal states, depicted as shaded cells.
- In each cell, the agent can move UP, RIGHT, DOWN, or LEFT.
- Actions deterministically move the agent in the specified direction, unless blocked by a wall.
- If the agent tries to move into a wall, it remains in its current position.
- The agent receives a reward of -1 for each step taken until it reaches a terminal state.

Policy Evaluation / Prediction

From Bellman's equation.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right]$$

We use iterative solution.

$$v_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_k(s') \right]$$

ITERATIVE POLICY EVALUATION

Input π, the policy to be evaluated and convergence threshold θ

Initialize state values $v(s) = 0$ or to any arbitrary values for all state s $\in$ S.
However terminal state values should always be initialized to 0

Make a copy: $v'(s) \leftarrow v(s)$ for all s

Loop:

$\Delta = 0$

Loop for each $s \in S$

$$v'(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$$

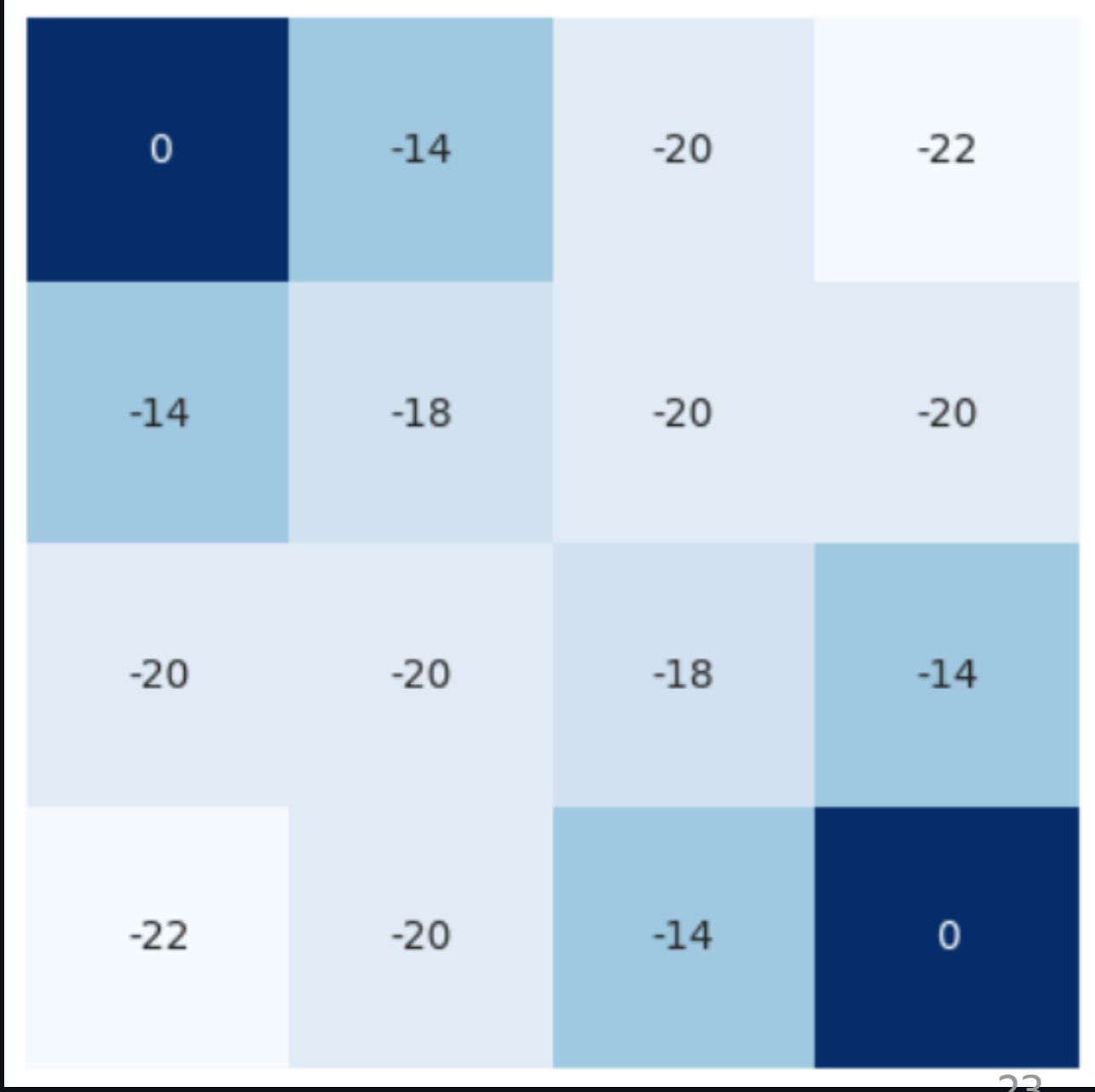
$$\Delta = max(\Delta, |v(s) - v'(s)|)$$

$v(s) \rightarrow v(s')$ for all $s \in S$, i.e., make a copy of $v(s)$

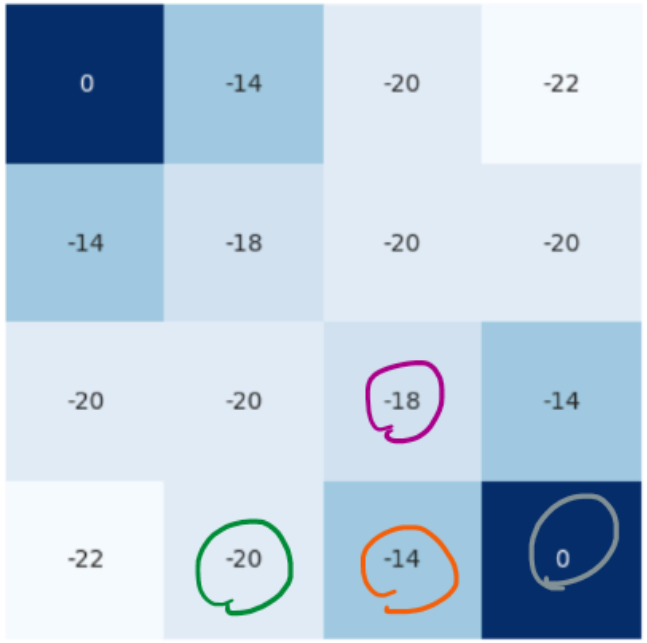
Until $\Delta < \theta$

Result

- See a python code that uses a random policy.



Validate Result

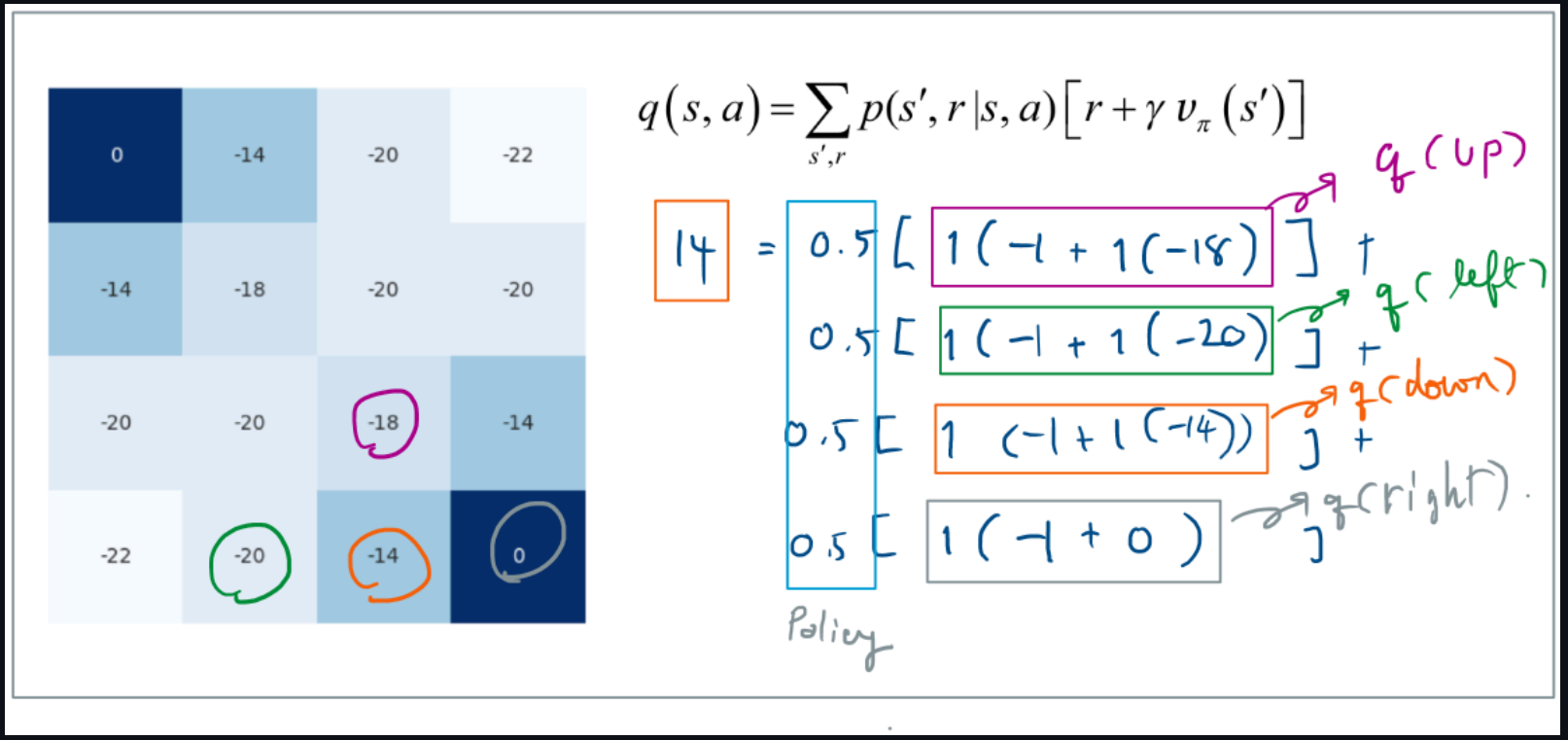


$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]$$

$14 = 0.5[1(-1 + 1(-18))] +$
$0.5[1(-1 + 1(-20))] +$
$0.5[1(-1 + 1(-14))] +$
$0.5[1(-1 + 0)]$

$14 = 0.5(-19 - 21 - 1 - 15) = 14$

Policy Improvement



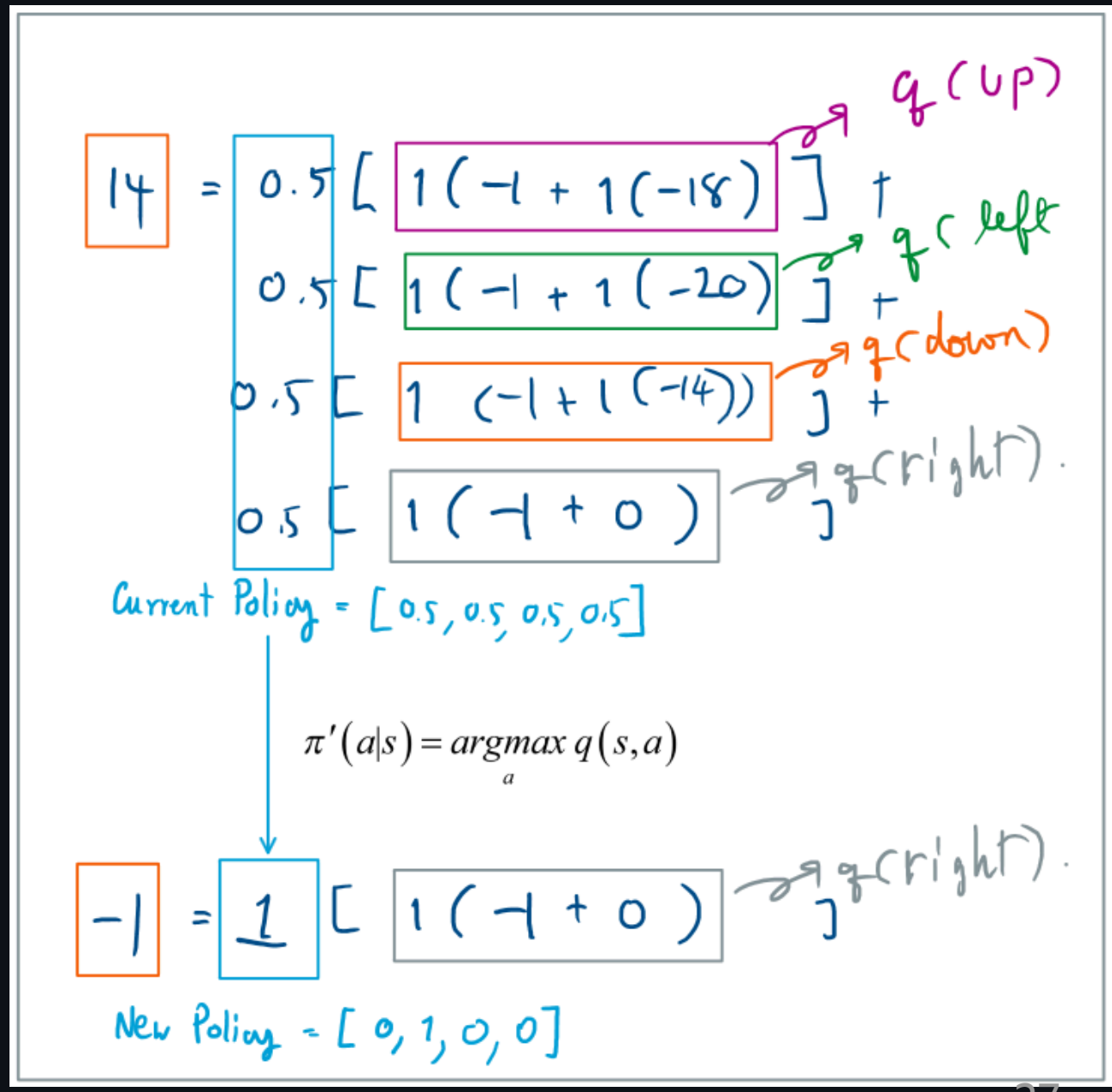
Policy Improvement

- You can improve the current policy by choose the action with maximum q value and define this to be the new policy.

$$\pi'(a|s) = \underset{a}{argmax}\, q(s,a)$$

Policy Improvement

- This is known as the *greedy* policy.
- Also Note the new state value $v = -1$.



Policy Iteration

$$\pi_0 \xrightarrow{evaluate} v_{\pi_0} \xrightarrow{improve} \pi_1 \xrightarrow{evaluate} v_{\pi_1} \xrightarrow{improve} \pi_2 \ldots\ldots \xrightarrow{improve} \pi_* \xrightarrow{evaluate} v_*$$

POLICY ITERATION

Initialize state values $v(s)$ and policy π arbitrarily
e.g. $v(s) = 0, \forall s \in S$, and $\pi(a|s)$ as random

define convergence threshold θ

Policy-Evaluation-Step

Loop:

$\Delta = 0$

Loop for each $s \in S$

$$v'(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$$

$$\Delta = \max(\Delta, |v(s) - v'(s)|)$$

$v(s) \leftarrow v'(s)$ for all $s \in S$, i.e. make a copy of $v(s)$

Until $\Delta < \theta$

Policy-Improvement-Step

policy-changed $\leftarrow$ false

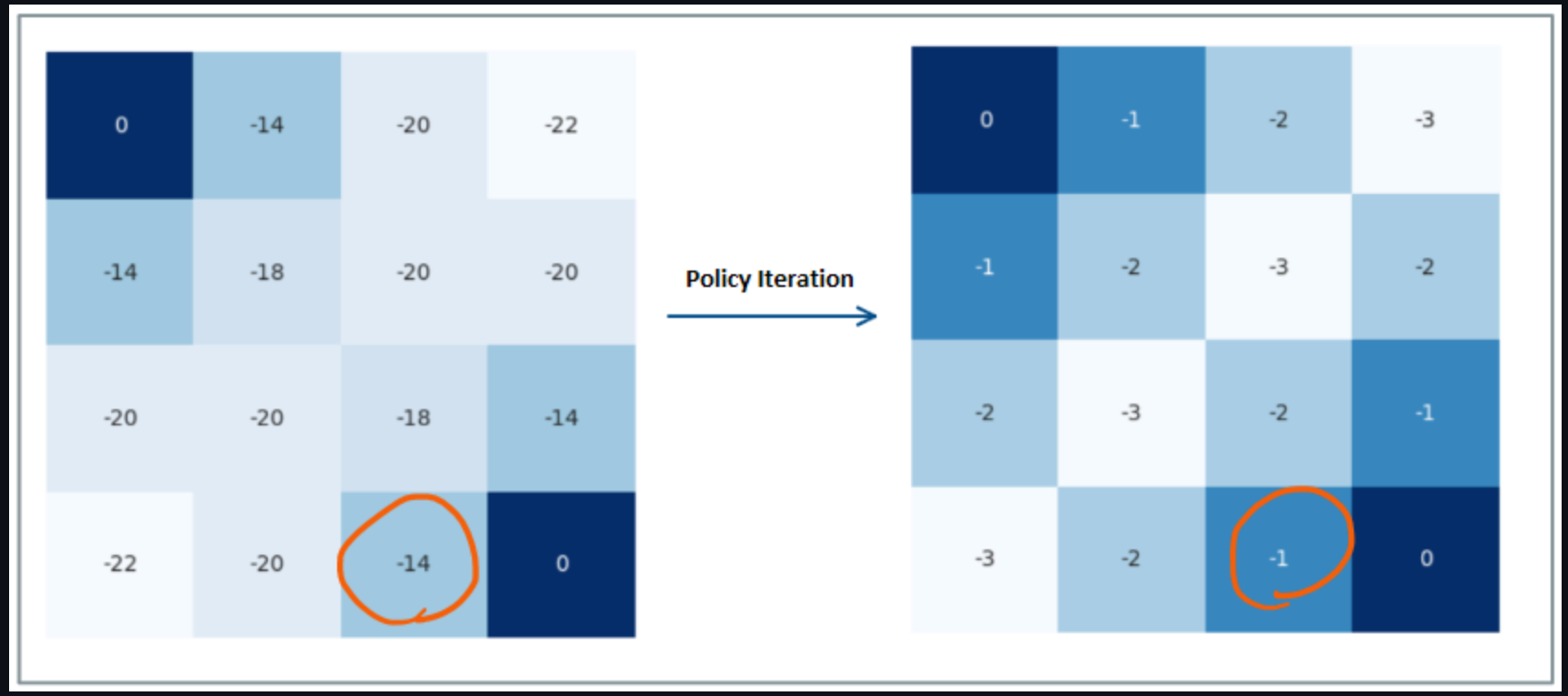
Loop for each $s \in S$

old-action $\leftarrow \pi(s)$

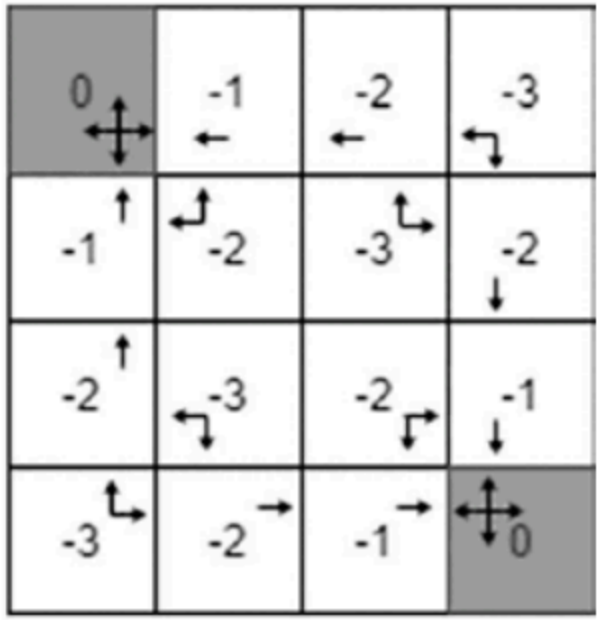
$$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$$

If $\pi(s) \neq$ old-action then policy-changed = true

If policy-changed = true: then go back to Policy-Evaluation step followe by Policy Improvement
otherwise return $v(s)$ as v_* and $\pi(s)$ as π_*

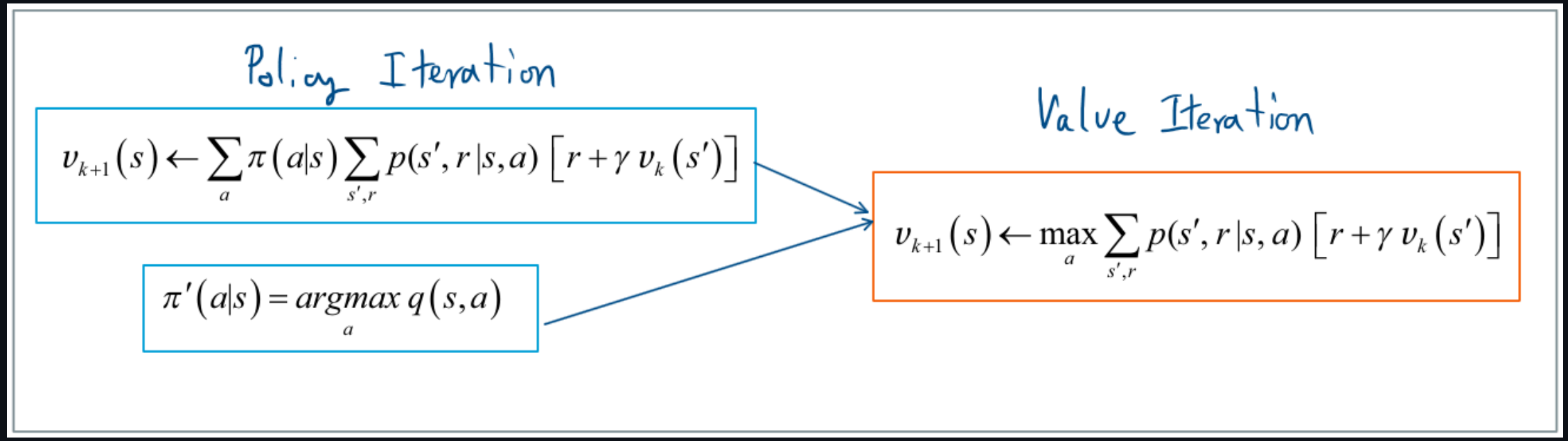

0 -14 -20 -22
-14 -18 -20 -20
-20 -20 -18 -14
-22 -20 -14 0
Policy Iteration
0 -1 -2 -3
-1 -2 -3 -2
-2 -3 -2 -1
-3 -2 -1 0

```
Optimal Policy
 [[0.25 0.25 0.25 0.25]
 [0.   0.   0.   1.  ]
 [0.   0.   0.   1.  ]
 [0.   0.   0.5  0.5 ]
 [1.   0.   0.   0.  ]
 [0.5  0.   0.   0.5 ]
 [0.   0.   0.5  0.5 ]
 [0.   0.   1.   0.  ]
 [1.   0.   0.   0.  ]
 [0.5  0.5  0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   0.   1.   0.  ]
 [1.   0.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.25 0.25 0.25 0.25]]
```



Value Iteration

- Merge the policy evaluation and policy improvement together into single operation.



Value Iteration

- The state values will keep improving.

$$v_0 \rightarrow v_1 \rightarrow \ldots v_k \rightarrow v_{k+1} \rightarrow \ldots \rightarrow v_*$$

- Once the values converges, we can find the optimum policy.

$$\pi_*(a|s) = \underset{a}{argmax} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s') \right]$$

VALUE ITERATION

Initialize state values $v(s)$ (Terminal states are always initialized to 0)
e.g. $v(s) = 0, \forall s \in S$
define convergence threshold θ
Make a copy: $v'(s) \leftarrow v(s)$ for all s

Loop:

$\Delta = 0$

Loop for each $s \in S$

$v'(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$

$\Delta = \max(\Delta, |v(s) - v'(s)|)$

$v(s) \leftarrow v'(s)$ for all $s \in S$, i.e. make a copy of $v(s)$

Until $\Delta < \theta$

Output a deterministic policy, breaking ties deterministically.
Initialize $\pi(s)$, an array of length $|S|$
Loop for each $s \in S$

$\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v(s')]$

Generalized Policy Iteration

- Main idea

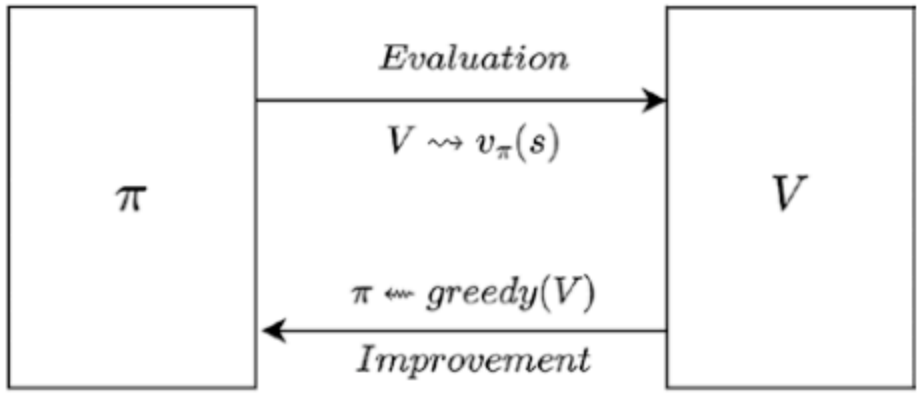


Figure 3-13. *Iteration between the two steps. The first step is evaluation to get state values in sync with the policy being followed. The second step is that of policy improvement to do a greedy maximization for actions*

Generalized Policy Iteration

- Full convergence + change policy for all states.

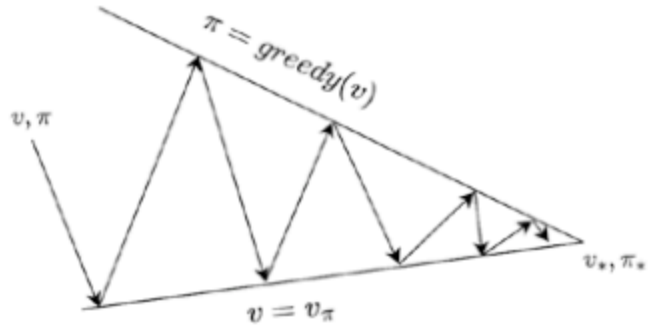


Figure 3-10. *Iteration between the two steps. The first step is evaluating to get the state values in sync with the policy being followed. The second step is improving the policy to do a greedy maximization for actions*

Generalized Policy Iteration

- **Partial** convergence + change policy for all states.

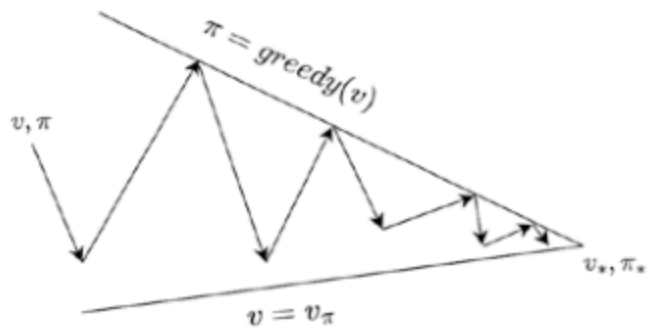


Figure 3-11. *Iterating just enough to cover all the states in a single iteration leads to not touching the bottom line* $v = v_\pi$

Generalized Policy Iteration

- Full convergence + change policy for **partial** states.

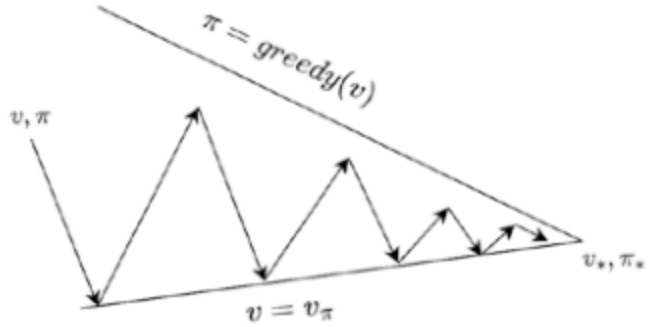


Figure 3-12. *Iterating when the policy improvement step is not carried out over all the states, leading to not touching the top line $\pi = greedy(v)$*