An example of an MDP

You are playing Simple Mario Party and given the following map:

Choose action = “up”: w.p. 0.6 → go up
w.p. 0.4 → go down

Choose action = “down”: w.p. 0.4 → go up
w.p. 0.6 → go down

- state: where you are > decide the distr. for next state
- action: up / down
- reward (per step): func. of state and action
  total reward: sum of rewards over the time horizon
- policy: which action to take in each state at each time step.
- goal: design a policy to maximize total expected reward

We next introduce the general form of an MDP, where we also generalize the time horizon from a finite one to an infinite one.
Markov decision processes (MDPs)

An MDP is a discrete-time process specified by:
- State space $S$. Let $s_t \in S$ be the state at time $t$.
- Action space $A$. Let $a_t \in A$: action at time $t$.

For simplicity, we focus on the setting where $S$ and $A$ are finite.

- Transition probabilities $P: S \times A \rightarrow \Delta(S)$, $\Delta(S)$: prob. distr. over $S$.
  $P(s', s, a)$: prob. of transitioning to state $s'$ when taking action $a$ in state $s$.

- Reward function: $r: S \times A \rightarrow \mathbb{R}$.
  $r(s, a)$: immediate reward when taking action $a$ in state $s$.
  $r(s, a)$ could be random, in which case $r(s, a)$ denotes its mean.

- Policy: In general, a policy can choose $a_t$ based on the full history
  $H_t = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$.

  We focus on stationary policies $\pi: S \rightarrow \Delta(A)$, which chooses actions based on only the current state, i.e., $a_t \sim \pi(\cdot | s_t)$.

  We sometimes simply write $a_t = \pi(s_t)$.

- Goal: Find a policy $\pi$ to solve

  $\max_{\pi} V^\pi(s) \triangleq \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi \right]$ 

  $\gamma \in (0, 1)$: discount factor.

Explanation: (1) $\gamma$: prob. that the problem continues after each time step.

(2) Reward now is more important than that in the future.

$V^\pi: S \rightarrow \mathbb{R}$: value function of policy $\pi$. 
Remark. In general, the optimization is over all policies, which can be non-stationary and non-Markovian. But it can be shown that optimality can be achieved by a stationary policy.
Bellman Equation (Dynamic Programming Equation)

Note that for a fixed policy $\pi$,

$$V^\pi(s) = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi \right]$$

$$= r(s, \pi(s)) + \sum_{s'} \mathbb{E}\left[ \sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \mid s_1 = s', \pi \right] \cdot \mathbb{P}(s' \mid s, \pi) \cdot r(s', \pi(s'))$$

$$= r(s, \pi(s)) + \gamma \mathbb{E} \left[ V^\pi(s_t) \mid s_0 = s, a_0 \sim \pi(\cdot \mid s) \right]$$

which gives an equation for $V^\pi$.

Let $V^*(s) = \sup_{\pi} V^\pi(s)$ be the optimal value function. Then $V^*$ satisfies a similar equation, referred to as the Bellman equation.

**Theorem.** The optimal value function $V^*$ satisfies

$$V^*(s) = \max_a \left( r(s, a) + \gamma \mathbb{E} \left[ V^*(s_t) \mid s_0 = s, a_0 = a \right] \right), \quad \forall s.$$  \hspace{1cm} (1)

Moreover, let policy $\pi^*$ be specified as

$$\pi^*(s) \in \arg\max_a \left( r(s, a) + \gamma \mathbb{E} \left[ V^*(s_t) \mid s_0 = s, a_0 = a \right] \right).$$ \hspace{1cm} (2)

Then $\pi^*$ is an optimal policy.

**Remark.** $\pi^*$ is a stationary, deterministic policy.
How can we make use of the Bellman equation to get an optimal policy? Naturally, we want to solve the Bellman equation to get \( V^* \), and then use equation (2) to get an optimal policy. To be able to do so, we need to answer the following questions:

(i) If we find a solution to the Bellman equation, is it guaranteed to be \( V^* \)?

(ii) How do we find a solution to the Bellman equation?

To answer both questions, it is convenient to write the Bellman equation using the so-called Bellman operator.

**Bellman Operator**

We index the state space as \( S = \{1, 2, \ldots, d \} \). Then a value function \( V \) can be written as a vector: \( V = (V(1), V(2), \ldots, V(d)) \in \mathbb{R}^d \).

Recall the Bellman equation:

\[
V(s) = \max_a \left( r(s, a) + \gamma \mathbb{E}[V(s') \mid s_0 = s, a_0 = a] \right), \quad \forall s \in S.
\]

We can rewrite the right-hand-side of the Bellman equation by defining the Bellman operator \( T : \mathbb{R}^d \rightarrow \mathbb{R}^d \), which takes a value function as input and outputs another value function. Specifically, for any \( V \in \mathbb{R}^d \), \( TV \in \mathbb{R}^d \) is defined as

\[
TV(s) = \max_a \left( r(s, a) + \gamma \mathbb{E}[V(s') \mid s_0 = s, a_0 = a] \right), \quad \forall s \in S.
\]

Then the Bellman equation can be written as: \( V = TV \).
Now let's return to the two questions:

(i) If we find a solution to the Bellman equation \( V = TV \), is it guaranteed to be \( V^* \)?

(ii) How do we find a solution to \( V = TV \)?

Solving \( V = TV \) is to find a fixed point of the operator \( T \). If \( T \) is a contraction mapping, then these two questions can be answered by the Banach fixed-point theorem.

**Contraction mapping:** Let \((X, d)\) be a complete metric space. Then a mapping \( T: X \to X \) is said to be a contraction mapping if there exists \( r \in (0, 1) \) such that \( d(T(x), T(y)) \leq r \cdot d(x, y) \), \( \forall x, y \in X \).

We say \( T \) has a fixed point \( x^* \) if \( T(x^*) = x^* \).

**Banach fixed-point theorem (contraction mapping theorem)**

Let \( T \) be a contraction mapping on a complete metric space \((X, d)\) with a contraction coefficient \( r \). Then

1. \( T \) has a unique fixed point \( x^* \).
2. The iterative algorithm \( x_{k+1} = T(x_k) \), starting from any initial point \( x_0 \in X \), has the property \( d(x_{k+1}, x^*) \leq r \cdot d(x_k, x^*) \).

As a result, \( x_k \to x^* \) geometrically fast, with the following equivalent descriptions of the convergence speed:

1. \( d(x_k, x^*) \leq r^k \cdot d(x_0, x^*) \)
2. \( d(x_k, x^*) \leq \frac{r^k}{1-r} \cdot d(x_0, x_0) \)

\[
( d(x_1, x_0) \geq d(x_0, x^*) - d(x_1, x^*) \geq (1-r) d(x_0, x^*) \\
\geq \frac{1-r}{1} d(x_k, x^*) )
\]
Is the Bellman operator a contraction mapping then?

**Theorem.** The Bellman operator \( T \) is a contraction mapping on \( \mathbb{R}^d \) under \( \| \cdot \|_{\infty} \) with the discount factor \( \gamma \) as a contraction coefficient, i.e., \( \forall V_1, V_2 \in \mathbb{R}^d, \| TV_1 - TV_2 \|_{\infty} \leq \gamma \| V_1 - V_2 \|_{\infty} \).

\[
\| x \|_{\infty} = \max \{ |x_1|, |x_2|, \ldots, |x_d| \}, \forall x \in \mathbb{R}^d
\]

**Proof.** Let \( s \in S \). Then

\[
TV_1(s) - TV_2(s) = \max_a \left( r(s, a) + \gamma \mathbb{E}[V_1(s') | s_0 = s, a_0 = a] \right)
- \max_{a'} \left( r(s, a') + \gamma \mathbb{E}[V_2(s') | s_0 = s, a_0 = a'] \right)
\]

\[
\leq r(s, a^*) + \gamma \mathbb{E}[V_1(s') | s_0 = s, a_0 = a^*]
- (r(s, a^*) + \gamma \mathbb{E}[V_2(s') | s_0 = s, a_0 = a^*])
\]

\[
= \gamma \mathbb{E}[V_1(s') - V_2(s') | s_0 = s, a_0 = a^*]
\]

\[
\leq \| V_1 - V_2 \|_{\infty}
\]

\[
\leq \gamma \| V_1 - V_2 \|_{\infty}
\]

Similarly, \( TV_2(x) - TV_1(x) \leq \gamma \| V_1 - V_2 \|_{\infty} \)

Therefore, \( \| TV_1 - TV_2 \|_{\infty} \leq \gamma \| V_1 - V_2 \|_{\infty} \). \( \square \)

**Implications.**

1. The Bellman equation \( V = TV \) has a unique solution.

   Therefore, the solution must be the optimal value function \( V^* \).

2. The iterative algorithm \( V_{k+1} = TV_k \) guarantees that \( V_k \rightarrow V^* \) as \( k \rightarrow \infty \).

   This gives rise to the value iteration algorithm below.
**Value iteration (VI)** Starting at some $V$, we iteratively apply $T: V \leftarrow TV$.

**Algorithm**
1. Initialize with a guess $V_0$, set $k = 0$.
2. $V_{k+1} = TV_k$
3. $k \leftarrow k+1$
4. Repeat 2-3 until "convergence".
5. Let $V_k$ be the output value function. Output policy $\pi_k$ defined by 
   \[ \pi_k(s) = \text{argmax}_a \left( r(s,a) + \gamma E[V_k(s_{i+1}) | s_0 = s, a_0 = a] \right), \]

From the contraction mapping theorem, we have convergence.

In practice, we need to use some stopping criterion.

If we stop after $k$ steps, how good is $V_k$ and how good is $\pi_k$?

- **Bound on $\|V_k - V^*\|_\infty$.** By the contraction mapping theorem,
  \[ \|V_k - V^*\|_\infty \leq \frac{\gamma^k}{1 - \gamma} \|V_1 - V_0\|_\infty. \]
  This bound is more useful than the bound $\|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty$ because $\|V_1 - V_0\|_\infty$ is computable while $\|V_0 - V^*\|_\infty$ is unknown.

How good is $\pi_k$? Note that $V_k$ is not necessarily the value function of $\pi_k$, but they are close. Recall that we use $V_{\pi_k}$ to denote the value function of $\pi_k$.

- **Bound on $\|V_{\pi_k} - V^*\|_\infty$.**
  \[ \|V_{\pi_k} - V^*\|_\infty \leq \|V_{\pi_k} - V_k\|_\infty + \|V_k - V^*\|_\infty. \]
  *just bounded*

Note that $V_{\pi_k}(s) = r(s, \pi_k(s)) + \gamma E[V_{\pi_k}(s_{i+1}) | s_0 = s, a_0 = \pi_k(s)]$
\[ = r(s, \pi_k(s)) + \gamma E[V_k(s_{i+1}) | s_0 = s, a_0 = \pi_k(s)] \]

$V_{\pi_k}(s)$ by definition of $\pi_k$
Thus \( \| V^\pi_k - V^\pi_{k+1} \|_\infty \leq \gamma \| V^\pi_k - V_k \|_\infty \).

We also know that \( \| V^\pi_k - V^\pi_{k+1} \|_\infty \geq \| V^\pi_k - V_k \|_\infty - \| V^\pi_{k+1} - V_k \|_\infty \).

So \( \| V^\pi_k - V_k \|_\infty \leq \frac{1}{1-\gamma} \| V^\pi_{k+1} - V_k \|_\infty \leq \frac{\gamma^k}{1-\gamma} \| V^* - V_0 \|_\infty \).

Putting them together, we have
\[
\| V^\pi_k - V^* \|_\infty \leq \| V^\pi_k - V_k \|_\infty + \| V_k - V^* \|_\infty \leq \frac{2\gamma^k}{1-\gamma} \| V^* - V_0 \|_\infty .
\]

The value iteration algorithm centers around the value function: it first makes sure that the value function obtained is close enough to the optimal value function, and then outputs a policy. Next we introduce another algorithm that promotes a more policy-centered view.
Policy iteration (PI). The structure of PI is as follows. We start from an arbitrary policy, and repeat the following iterative procedure:

1. Policy evaluation: calculate the value function of the policy.
2. Policy improvement: update the policy to improve it.

To make these two steps more concrete, we first define the operator associated with a policy for convenience. When we fix a policy $\pi$, we know that its value function $V^\pi$ satisfies

$$V^\pi(s) = r(s, \pi(s)) + rE[V^\pi(s') \mid s_0 = s, a_0 = \pi(s)], \forall s \in S.$$ 

Similar to the Bellman operator, the operator $T^\pi$ associated with policy $\pi$ is defined based on the right-hand-side of the equation. Specifically, for any $V \in \mathbb{R}^d$, $T^\pi V \in \mathbb{R}^d$ is defined as

$$T^\pi V(s) = r(s, \pi(s)) + rE[V^\pi(s') \mid s_0 = s, a_0 = \pi(s)], \forall s \in S.$$ 

Then the equation for policy $\pi$ can be written as: $V^\pi = T^\pi V^\pi$.

Note that $T^\pi$ is a linear operator.

Claim. $T^\pi$ is a contraction mapping on $\mathbb{R}^d$ under $\| \cdot \|_\infty$ with the discount factor $r$ as a contraction coefficient, i.e., for $V_1, V_2 \in \mathbb{R}^d$,

$$\| T^\pi V_1 - T^\pi V_2 \|_\infty \leq r \| V_1 - V_2 \|_\infty.$$

Implication. 1. The equation $V = T^\pi V$ has a unique solution, which is the value function of $\pi$, $V^\pi$.

2. In the policy evaluation step, we can use the iterative algorithm $V_{k+1} = T^\pi V_k$ to calculate $V^\pi$.

We can also show that both the Bellman operator $T$ and the operator $T^\pi$ are monotonic, i.e., $V_1 \leq V_2 \Rightarrow TV_1 \leq TV_2$, $T^\pi V_1 \leq T^\pi V_2$. 
The policy improvement step.

We can improve a policy using the right-hand-side of the Bellman equation. To update the policy \( \pi_k \) at the kth iteration, we define \( \pi_{k+1} \) as

\[
\pi_{k+1}(s) \in \arg\max_a \left( r(x,a) + \gamma E\left[ V^\pi_k(s') | s_0 = s, a_0 = a \right] \right), \forall s.
\]

Using the notation of operators, this implies that

\[
T^{\pi_{k+1}} V^\pi_k = TV^\pi_k.
\]

Putting the two steps together, the PI algorithm is given by:

1. Start with a policy \( \pi_0 \). Set \( k = 0 \).
2. Compute the value function \( V^\pi_k \) of policy \( \pi_k \) using the equation \( V = T^{\pi_k} V \).
3. Update the policy:

\[
\pi_{k+1}(s) \in \arg\max_a \left( r(x,a) + \gamma E\left[ V^\pi_k(s') | s_0 = s, a_0 = a \right] \right), \forall s.
\]
4. \( k \leftarrow k+1 \)
5. Repeat 2–4 until "convergence".

**Theorem** Under policy iteration, we have

1. \( V^{\pi_{k+1}} \geq V^{\pi_k} \), i.e., the policy improves at each step, and
2. If \( V^{\pi_{k+1}} = V^{\pi_k} \), then \( \pi_k \) is an optimal policy.

**Proof** (1) By step 3, \( T^{\pi_{k+1}} V^{\pi_k} = TV^{\pi_k} \geq T^{\pi_k} V^{\pi_k} = V^{\pi_k} \).

By the monotonicity of \( T^{\pi_{k+1}} \), we have

\[
T^{\pi_{k+1}} (T^{\pi_{k+1}} V^{\pi_k}) \geq T^{\pi_{k+1}} (V^{\pi_k}) \leq V^{\pi_k}
\]

Keep applying \( T^{\pi_{k+1}} \) \( N \) times, we get

\[
(T^{\pi_{k+1}})^N V^{\pi_k} \geq V^{\pi_k}.
\]

By the contraction property of \( T^{\pi_{k+1}} \), taking \( N \to \infty \) gives

\[
V^{\pi_{k+1}} \geq V^{\pi_k} \]

---

Under policy iteration, we have

1. \( V^{\pi_{k+1}} \geq V^{\pi_k} \), i.e., the policy improves at each step, and
2. If \( V^{\pi_{k+1}} = V^{\pi_k} \), then \( \pi_k \) is an optimal policy.

**Proof** (1) By step 3, \( T^{\pi_{k+1}} V^{\pi_k} = TV^{\pi_k} \geq T^{\pi_k} V^{\pi_k} = V^{\pi_k} \).

By the monotonicity of \( T^{\pi_{k+1}} \), we have

\[
T^{\pi_{k+1}} (T^{\pi_{k+1}} V^{\pi_k}) \geq T^{\pi_{k+1}} (V^{\pi_k}) \leq V^{\pi_k}
\]

Keep applying \( T^{\pi_{k+1}} \) \( N \) times, we get

\[
(T^{\pi_{k+1}})^N V^{\pi_k} \geq V^{\pi_k}.
\]

By the contraction property of \( T^{\pi_{k+1}} \), taking \( N \to \infty \) gives

\[
V^{\pi_{k+1}} \geq V^{\pi_k} \]
(2) If $V^{\pi_{k+1}} = V^{\pi_k}$, then $T^{\pi_{k+1}} V^{\pi_k} = T V^{\pi_k} \Rightarrow T^{\pi_{k+1}} V^{\pi_{k+1}} = T V^{\pi_{k+1}}$
$\Rightarrow V^{\pi_{k+1}} = T V^{\pi_{k+1}}$. So $V^{\pi_{k+1}}$ satisfies the Bellman equation, which means that $\pi_{k+1}$ and $\pi_k$ are optimal policies.

**Implications of the theorem.** The theorem says that at each step, you either get an improved policy or you have found the optimal policy.

- So in principle, PI converges in a finite number of steps when the state space and action space are finite.

- However, in each step, one needs to compute $V^{\pi_k}$. This can be done using the iterative algorithm $V_{i+1} = T^{\pi_k} V_i$. This inner loop can take a long time to produce an accurate value for $V^{\pi_k}$. 
Recall the Bellman equation:

\[ V^*(s) = \max_a \left( r(s, a) + \gamma \sum_{s'} V^*(s') \cdot P(s'|s, a) \right) \]

Suppose \( V^* \) is known, we still cannot solve this maximization problem to get the optimal policy without knowing the model \( P(s'|s, a) \). However, if we obtain the following function

\[ Q(s, a) = r(s, a) + \gamma \sum_{s'} V^*(s') \cdot P(s'|s, a), \]

then we can solve \( \max_a Q(s, a) \) to get the optimal policy. The function \( Q : S \times A \rightarrow \mathbb{R} \) is called the (optimal) Q-function.

Meaning of \( Q(s, a) \): the total discounted reward when we take action \( a \) in the current step and follow the optimal policy in all the future time steps.

How can we get the Q-function? A starting point is the equation below derived from the Bellman equation. Note that \( V^*(s) = \max_a Q(s, a) \).

Then

\[
Q(s, a) = r(s, a) + \gamma \sum_{s'} V^*(s') \cdot P(s'|s, a) \\
= r(s, a) + \gamma \sum_{s'} P(s'|s, a) \cdot \max_{a'} Q(s', a').
\]

Directly evaluating the right-hand-side still requires the knowledge of \( P(s'|s, a) \), but there are many ways to learn the Q-function when the model is unknown.

We can also define the Q-function for a fixed policy \( \pi \) as follows:

\[ Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}[V^\pi(s_t)|s_0 = s, a_0 = a]. \]

This is the total discounted reward when we take action \( a \) in the current step.
time step and follow the policy \( \pi \) in the future. Then
\[
V^\pi(s) = E[Q^\pi(s, a) | a \sim \pi(s)].
\]
In many RL approaches, we need to evaluate the \( Q \)-function for a
given policy \( \pi \).