Probabilistic planning with full observability

- Several algorithms:
  1. dynamic programming (finite horizons)
  2. value iteration (discounted rewards, infinite horizon)
  3. policy iteration (discounted rewards, infinite horizon)

- Some of these algorithms can be easily implemented without explicitly representing the state space (e.g. by using algebraic decision diagrams ADDs).

Optimal plans over a finite horizon

The optimum values \( v_t(s) \) for states \( s \in S \) at time \( t \in \{1, \ldots, N\} \) fulfill the following equations. (\( A(s) = \) actions applicable in \( s \))

\[
\begin{align*}
v_N(s) &= \max_{a \in A(s)} R(s, a) \\
v_i(s) &= \max_{a \in A(s)} \left( R(s, a) + \sum_{s' \in S} p(s' | s, a) v_{i+1}(s') \right) \text{ for } i \in \{1, \ldots, N-1\}
\end{align*}
\]

Optimal plans over a finite horizon: plans

Action for state \( s \in S \) at time \( t \) is \( \pi(s, t) \) defined by

\[
\pi(s, N) = \arg \max_{a \in A(s)} R(s, a)
\]

\[
\pi(s, i) = \arg \max_{a \in A(s)} \left( R(s, a) + \sum_{s' \in S} p(s' | s, a) v_{i+1}(s') \right) \text{ for } i \in \{1, \ldots, N-1\}
\]

If plan execution is actually unbounded, best action is \( \pi(s, 1) \)!
(receding-horizon control)
An example for demonstrating the algorithms

```
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>p=0.9</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p=0.1</td>
<td>0.1</td>
<td>0</td>
<td>0.9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R=5</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>p=0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
</tr>
<tr>
<td>R=1</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Total rewards with finite horizon: example

```
<table>
<thead>
<tr>
<th>i</th>
<th>v_i(A)</th>
<th>v_i(B)</th>
<th>v_i(C)</th>
<th>v_i(D)</th>
<th>v_i(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>5.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>4.6</td>
<td>1.0</td>
<td>5.0</td>
<td>1.0</td>
</tr>
<tr>
<td>7</td>
<td>4.6</td>
<td>4.6</td>
<td>1.0</td>
<td>6.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
```
### Total rewards with finite horizon: example

<table>
<thead>
<tr>
<th>i</th>
<th>(v_i(A))</th>
<th>(v_i(B))</th>
<th>(v_i(C))</th>
<th>(v_i(D))</th>
<th>(v_i(E))</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>5.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>4.6</td>
<td>1.0</td>
<td>5.0</td>
<td>1.0</td>
</tr>
<tr>
<td>7</td>
<td>4.6</td>
<td>4.6</td>
<td>1.0</td>
<td>6.0</td>
<td>1.0</td>
</tr>
<tr>
<td>6</td>
<td>4.6</td>
<td>5.86</td>
<td>4.6</td>
<td>6.0</td>
<td>4.6</td>
</tr>
<tr>
<td>5</td>
<td>5.86</td>
<td>5.86</td>
<td>4.6</td>
<td>9.6</td>
<td>4.6</td>
</tr>
<tr>
<td>4</td>
<td>5.86</td>
<td>9.23</td>
<td>5.86</td>
<td>9.6</td>
<td>5.86</td>
</tr>
<tr>
<td>3</td>
<td>9.23</td>
<td>9.23</td>
<td>5.86</td>
<td>10.86</td>
<td>5.86</td>
</tr>
<tr>
<td>2</td>
<td>9.23</td>
<td>10.70</td>
<td>9.23</td>
<td>10.86</td>
<td>9.23</td>
</tr>
<tr>
<td>1</td>
<td>10.70</td>
<td>10.70</td>
<td>9.23</td>
<td>14.23</td>
<td>9.23</td>
</tr>
</tbody>
</table>

### Optimality / Bellman equations (infinite horizon)

Values \(v(s)\) of states \(s \in S\) are the discounted sum of the expected rewards obtained by choosing the best possible actions in \(s\) and in its successors.

\[
v(s) = \max_{a \in A(s)} \left( R(s, a) + \sum_{s' \in S} \lambda p(s'|s, a) v(s') \right)
\]  \hspace{1cm} (1)

\(\lambda\) is the discount constant: \(0 < \lambda < 1\).

### Plan evaluation by solving linear equations

Given a plan \(\pi\), its value under discounted rewards with discount constant \(\lambda\) satisfies the following equation for every \(s \in S\).

\[
v(s) = R(s, \pi(s)) + \sum_{s' \in S} \lambda p(s'|s, \pi(s)) v(s')
\]  \hspace{1cm} (2)

This yields a system of \(|S|\) linear equations and \(|S|\) unknowns. The solution of these equations gives the value of the plan in each state.

### Plan evaluation: example

Consider the plan

\[
\pi(A) = R, \pi(B) = R, \pi(C) = B, \pi(D) = R, \pi(E) = B
\]

\[
v_{\pi}(A) = R(A, R) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(C) + 0 \lambda v_{\pi}(D) + 0 \lambda v_{\pi}(E)
\]
\[
v_{\pi}(B) = R(B, R) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(C) + 0 \lambda v_{\pi}(D) + 0 \lambda v_{\pi}(E)
\]
\[
v_{\pi}(C) = R(C, B) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(B) + 0 \lambda v_{\pi}(C) + 0 \lambda v_{\pi}(D) + 0 \lambda v_{\pi}(E)
\]
\[
v_{\pi}(D) = R(D, R) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(C) + 0 \lambda v_{\pi}(D) + 0 \lambda v_{\pi}(E)
\]
\[
v_{\pi}(E) = R(E, B) + 0 \lambda v_{\pi}(A) + 0 \lambda v_{\pi}(B) + 0 \lambda v_{\pi}(C) + 0 \lambda v_{\pi}(D) + 0 \lambda v_{\pi}(E)
\]
\begin{align*}
v_\pi(A) &= 1 + \lambda v_\pi(C) \\
v_\pi(B) &= 0 + 0.1 \lambda v_\pi(A) + 0.9 \lambda v_\pi(D) \\
v_\pi(C) &= 0 + \lambda v_\pi(E) \\
v_\pi(D) &= 5 + \lambda v_\pi(E) \\
v_\pi(E) &= 0 + \lambda v_\pi(C) \\

v_\pi(A) - \lambda v_\pi(C) &= 1 \\
-0.1 \lambda v_\pi(A) + v_\pi(B) - 0.9 \lambda v_\pi(D) &= 0 \\
v_\pi(C) - \lambda v_\pi(E) &= 0 \\
v_\pi(D) - \lambda v_\pi(E) &= 5 \\
- \lambda v_\pi(C) + v_\pi(E) &= 0
\end{align*}

Solving with $\lambda = 0.5$ we get

\begin{align*}
v_\pi(A) &= 1 \\
v_\pi(B) &= 2.3 \\
v_\pi(C) &= 0 \\
v_\pi(D) &= 5 \\
v_\pi(E) &= 0
\end{align*}

This is the value function of the plan.

---

**The Value Iteration algorithm**

- **Value Iteration** is the simplest algorithm for finding plans that are close to optimal (discounted rewards, infinite horizon).

- **Idea:**
  1. Start with an arbitrary value function.
  2. Compute better and better approximations of the optimal value function.
  3. From a good approximation construct a plan.

---

**The Value Iteration algorithm: convergence**

- Optimal value function is never reached.

- Plans extracted from a very-close-to-optimal value function are typically optimal.

- Important parameter $\epsilon$:
  Algorithm terminates when change in the value function was smaller than $\frac{(1-\lambda)}{2\lambda}$.

Then distance from optimal value function is smaller than $\epsilon$. 
The Value Iteration algorithm: definition

1. \( n := 0 \)
2. Assign initial values to \( v^0(s) \) for all \( s \in S \).
3. \[
v^{n+1}(s) := \max_{a \in A(s)} \left( R(s, a) + \sum_{s' \in S} \lambda p(s'|s, a)v^n(s') \right) \text{ for every } s \in S
\]

If \(|v^{n+1}(s) - v^n(s)| < \frac{(1-\lambda)^n}{2n} \) for all \( s \in S \) then go to step 4.
Otherwise, set \( n := n + 1 \) and go to step 3.

4. Construct a plan: for every \( s \in S \)
   \[
   \pi(s) := \arg \max_{a \in A(s)} \left( R(s, a) + \sum_{s' \in S} \lambda p(s'|s, a)v^{n+1}(s') \right)
   \]

The Value Iteration algorithm: properties

THEOREM Let \( v^*_\pi \) be the value function of the plan produced by the value iteration algorithm, and let \( v^* \) be the value function of the optimal plan \( \pi(s) \). Then \(|v^*(s) - v^*_\pi(s)| \leq \epsilon\) for all \( s \in S \).

Under full observability there is never a trade-off between the values of two states: if the optimal value for state \( s_1 \) is \( r_1 \) and the optimal value for state \( s_2 \) is \( r_2 \), then there is one plan that achieves these both.
The Policy Iteration algorithm

- Finds optimal plans.
- Slightly more complicated to implement: each iteration consists of
  - evaluation of the value of the best plan so far, and
  - improving the plan.
- Number of iterations is small.
- Runtimes often higher than with value iteration.

The Policy Iteration algorithm: definition

1. Assign $n := 0$.
2. Let $\pi^0$ be any mapping from states $s \in S$ to actions in $A(s)$.
3. Compute the values $v^n(s)$ of all $s \in S$ under $\pi^n$.
4. Let $\pi^{n+1}(s) = \arg\max_{a \in A(s)} \left( R(s, a) + \sum_{s' \in S} \lambda p(s'|s, a) v^n(s') \right)$.
5. Assign $n := n + 1$.
6. If $n = 1$ or $v^n \neq v^{n-1}$ then go to 3.

The Policy Iteration algorithm: properties

THEOREM The policy iteration algorithm terminates after a finite number of steps and returns an optimal plan.

PROOF IDEA: There is only a finite number of different plans, and at each step a properly better plan is found or the algorithm terminates.

Policy iteration: example

<table>
<thead>
<tr>
<th>itr</th>
<th>$\pi(A)$</th>
<th>$\pi(B)$</th>
<th>$\pi(C)$</th>
<th>$\pi(D)$</th>
<th>$\pi(E)$</th>
<th>$v_n(A)$</th>
<th>$v_n(B)$</th>
<th>$v_n(C)$</th>
<th>$v_n(D)$</th>
<th>$v_n(E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>1.56</td>
<td>3.09</td>
<td>0.93</td>
<td>5.56</td>
<td>0.93</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>1.91</td>
<td>3.18</td>
<td>1.14</td>
<td>5.68</td>
<td>1.14</td>
</tr>
</tbody>
</table>

The number of iterations needed for finding an $\epsilon$-optimal plan by policy iteration is never higher than the number of iterations needed by value iteration.