Planning & Decision-making in Robotics

Planning under Uncertainty: Expected Formulation, Solving MDPs

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Minimax Formulation is Often Too Conservative

Example:

moving over the hill has 10% chance of slipping
Expected Cost Formulation

\[ c(s_1, a_1, s_2) = 2 \]
\[ P(s_2 | s_1, a_1) = 0.1 \]
\[ P(s_{goal} | s_1, a_1) = 0.9 \]
\[ c(s_1, a_1, s_{goal}) = 2 \]

- Optimal policy \( \pi^* \):
  minimizes the expected cost-to-goal
  \[ \pi^* = \arg\min_\pi E\{\text{cost-to-goal}\} \]
Expected Cost Formulation

- Optimal policy $\pi^*$: minimizes the expected cost-to-goal
  $$\pi^* = \text{argmin}_\pi E\{\text{cost-to-goal}\}$$

- expected cost-to-goal for $\pi_1 =$ (go through $s_4$) is
  $$1 + 1 + 3 + 1 = 6$$

- cost-to-goal for $\pi_2 =$ (try to go through $s_1$) is:
  $$0.9 \times (1 + 2 + 2) + 0.9 \times 0.1 \times (1 + 2 + 2 + 2 + 2) + 0.9 \times 0.1 \times 0.1 \times (1 + 2 + 2 + 2 + 2 + 2 + 2) + \ldots = 5.444$$
Expected Cost Formulation

• Optimal policy $\pi^*$: minimizes the expected cost-to-goal

$\pi^* = \text{argmin}_\pi E\{\text{cost-to-goal}\}$

• expected cost-to-goal for $\pi_1 = (\text{go through } s_4)$:

$1 + 1 + 3 + 1 = 6$

• cost-to-goal for $\pi_2 = (\text{try to go through } s_1)$ is:

$0.9(1+2+2) + 0.9 \times 0.1(1+2+2+2+2) + 0.9 \times 0.1 \times 0.1(1+2+2+2+2+2) + \ldots = 5.444$

How to compute it?

Given a policy, its value can be computed by solving a system of linear equations expectation over outcomes.
Expected Cost Formulation

\[ c(s_1, a_1, s_{\text{goal}}) = 2 \]

\[ P(s_{\text{goal}} | s_1, a_1) = 0.9 \]

\[ P(s_2 | s_1, a_1) = 0.1 \]

\[ S_{\text{start}} \]
\[ S_2 \]
\[ S_1 \]
\[ a_1 \]
\[ S_3 \]
\[ S_4 \]
\[ S_{\text{goal}} \]

- Optimal policy \( \pi^* \): minimizes the expected cost-to-goal.

Given a policy, its value can be computed by solving a system of linear equations.

- expected cost-to-goal for \( \pi_1 \):
  \[ v(s_{\text{start}}) = 1 + v(s_2) \]
  \[ v(s_2) = 2 + v(s_1) \]
  \[ v(s_1) = 0.9(2 + v(s_{\text{goal}})) + 0.1(2 + v(s_2)) \]
  \[ v(s_{\text{goal}}) = 0 \]

- cost-to-goal for \( \pi_2 \) = (try to go through \( S_1 \))
  \[ 0.9 + 1 + 3 + 1 = 6 \]
  \[ 0.9(1 + 2 + 2) + 0.9(0.1(1 + 2 + 2 + 2 + 2)) + 0.9(0.1(1 + 2 + 2 + 2 + 2 + 2 + 2)) + \ldots = 5.444 \]
Expected Cost Formulation

- Optimal policy $\pi^*$: minimizes the expected cost-to-goal
  $\pi^* = \text{argmin}_\pi E\{\text{cost-to-goal}\}$

- Optimal expected cost policy $\pi^* = \pi_2 = (\text{go through } s_1)$
Expected Cost Formulation

\[ c(s_1, a_1, s_{goal}) = 2 \]

\[ P(s_{goal}|s_1, a_1) = 0.9 \]

\[ c(s_1, a_1, s_2) = 2 \]

\[ P(s_2|s_1, a_1) = 0.1 \]

• Optimal policy \( \pi^* \):
  
  minimizes the expected cost-to-goal
  
  \( \pi^* = argmin_\pi E\{cost-to-goal\} \)

• Optimal expected cost policy \( \pi^* = \pi_2 = (go \ through \ s_1) \)

In contrast, optimal policy for minimax formulation was \( \pi_1 = (go \ through \ s_4) \)
Computing Expected Cost Minimal Plans

• Optimal policy $\pi^*$:
  minimizes the expected cost-to-goal
  $\pi^* = \arg\min_\pi E\{\text{cost-to-goal}\}$

• Let $\nu^*(s)$ be minimal expected cost-to-goal for state $s$
Computing Expected Cost Minimal Plans

- Optimal policy $\pi^*$:
  $$\pi^*(s) = \arg\min_a E\{c(s, a, s') + v^*(s')\}$$
  (expectation over outcomes $s'$ of action $a$ executed at state $s$)

Why?
Computing Expected Cost Minimal Plans

• Optimal expected cost-to-goal values $v^*$ satisfy:

$$v^*(s_{\text{goal}}) = 0$$
$$v^*(s) = \min_a E\{c(s,a,s') + v^*(s')\} \text{ for all } s \neq s_{\text{goal}}$$

(Expectation over outcomes $s'$ of action $a$ executed at state $s$)

Bellman optimality equation
• Value Iteration (VI):

  Initialize $v$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$
  $v(s) = \min_a E\{c(s,a,s') + v(s')\}$ for any $s \neq s_{goal}$
Computing Expected Cost Minimal Plans

• Value Iteration (VI):
  Initialize $v$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:
  \[
  v(s_{\text{goal}}) = 0
  \]
  \[
  v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}}
  \]

Bellman update equation (or backup)
Computing Expected Cost Minimal Plans

• Value Iteration (VI):
  
  Initialize \( v \)-values of all states to finite values;
  
  Iterate over all \( s \) in MDP and re-compute until convergence:
  
  \[
  v(s_{\text{goal}}) = 0
  \]
  
  \[
  v(s) = \min_a \mathbb{E}\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}}
  \]

  
  converges to an optimal value function
  \( (v(s) = v^*(s) \text{ for all } s) \)
  
  for any iteration order

  the speed of convergence
  depends on iteration order

  best to initialize to admissible values
  (under-estimates of the actual costs-to-goal)
Computing Expected Cost Minimal Plans

Value Iteration (VI):

- Initialize $v$-values of all states to finite values;
- Iterate over all $s$ in MDP and re-compute until convergence:
  
  $v(s_{goal}) = 0$
  
  $v(s) = \min_a E\{c(s,a,s') + v(s')\}$ for any $s \neq s_{goal}$

Best to initialize to admissible values (under-estimates of the actual costs-to-goal)

Converges to an optimal value function ($v(s) = v^*(s)$ for all $s$) for any iteration order

The speed of convergence depends on iteration order

Any ideas for the order?
Computing Expected Cost Minimal Plans

- **Value Iteration (VI):**
  Initialize \( v \)-values of all states to finite values;
  Iterate over all \( s \) in MDP and re-compute until convergence:

  \[
  v(s_{goal}) = 0 \\
  v(s) = \min_a \mathbb{E}\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{goal}
  \]
Computing Expected Cost Minimal Plans

- Value Iteration (VI):
  
  Initialize $v$-values of all states to finite values;
  
  Iterate over all $s$ in MDP and re-compute until convergence:
  
  $v(s_{goal}) = 0$
  
  $v(s) = \min_a E\{c(s,a,s') + v(s')\}$ for any $s \neq s_{goal}$
Computing Expected Cost Minimal Plans

• Value Iteration (VI):
  Initialize $v$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:
  \[ v(s_{\text{goal}}) = 0 \]
  \[ v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}} \]
Computing Expected Cost Minimal Plans

- Value Iteration (VI):
  Initialize $v$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:
  
  $v(s_{goal}) = 0$
  
  $v(s) = min_a E\{c(s,a,s') + v(s')\}$ for any $s \neq s_{goal}$

After backing up $s_3$
Computing Expected Cost Minimal Plans

Value Iteration (VI):

Initialize $v$-values of all states to finite values;
Iterate over all $s$ in MDP and re-compute until convergence:

$v(s_{\text{goal}}) = 0$

$v(s) = \min_a E\{c(s,a,s') + v(s')\}$ for any $s \neq s_{\text{goal}}$
Computing Expected Cost Minimal Plans

• Value Iteration (VI):
  Initialize $\nu$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:
  
  $$\nu(s_{goal}) = 0$$
  $$\nu(s) = \min_a E\{c(s,a,s') + \nu(s')\} \text{ for any } s \neq s_{goal}$$

  Usual convergence condition: Bellman error over all states $< \Delta$
  Bellman error: $|\nu(s) - \min_a E\{c(s,a,s') + \nu(s')\}|$ for any $s \neq s_{goal}$
Computing Expected Cost Minimal Plans

\[ c(s_1, a_1, s_{goal}) = 2 \]

\[ P(s_{goal} | s_1, a_1) = 0.9 \]

\[ c(s_1, a_1, s_2) = 2 \]

\[ P(s_2 | s_1, a_1) = 0.1 \]

\[ v(s_{start}) = 2 \]

\[ v(s_2) = 1 \]

\[ v(s_1) = 2.1 \]

\[ v(s_{goal}) = 0 \]

\[ v(s_3) = 1 \]

\[ v(s_4) = 4 \]

\[ v = 0 \] after backing up \( s_1 \)

### Value Iteration (VI):

- Initialize \( v \)-values of all states to finite values;
- Iterate over all \( s \) in MDP and re-compute until convergence:

\[
v(s_{goal}) = 0
\]

\[
v(s) = \min_{a} E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{goal}
\]

Usual convergence condition: Bellman error over all states < \( \Delta \)

Bellman error: \(|v(s) - \min_{a} E\{c(s,a,s') + v(s')\}| \text{ for any } s \neq s_{goal}\)
Computing Expected Cost Minimal Plans

- Value Iteration (VI):
  Initialize \( v \)-values of all states to finite values;
  Iterate over all states in MDP and re-compute until convergence:
  
  \[
  v(s_{\text{goal}}) = 0
  \]
  \[
  v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}}
  \]

  Usual convergence condition: Bellman error over all states < \( \Delta \)

  Bellman error: \( |v(s) - \min_a E\{c(s,a,s') + v(s')\}| \) for any \( s \neq s_{\text{goal}} \)
Computing Expected Cost Minimal Plans

- Value Iteration (VI):
  
  Initialize $v$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:
  
  \[
  v(s_{goal}) = 0 \\
  v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{goal}
  \]

  Usual convergence condition: Bellman error over all states $< \Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s') + v(s')\}|$ for any $s \neq s_{goal}$
Computing Expected Cost Minimal Plans

- Value Iteration (VI):
  
  Initialize $v$-values of all states to finite values;
  
  Iterate over all $s$ in MDP and re-compute until convergence:
  
  $$v(s_{\text{goal}}) = 0$$
  $$v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}}$$

  Usual convergence condition: Bellman error over all states $< \Delta$
  
  Bellman error: $|v(s) - \min_a E\{c(s,a,s') + v(s')\}|$ for any $s \neq s_{\text{goal}}$
Computing Expected Cost Minimal Plans

- **Value Iteration (VI):**
  
  Initialize \( v \)-values of all states to finite values;
  
  Iterate over all \( s \) in MDP and re-compute until convergence:
  
  \[
  v(s_{\text{goal}}) = 0 \\
  v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}}
  \]

  *Usual convergence condition: Bellman error over all states < \( \Delta \)*

  *Bellman error: \( |v(s) - \min_a E\{c(s,a,s') + v(s')\}| \text{ for any } s \neq s_{\text{goal}} \)*
Computing Expected Cost Minimal Plans

\begin{itemize}
  \item Value Iteration (VI):
    \begin{align*}
    &v(s) = \min_{a} E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}} \\
    &v(s_{\text{goal}}) = 0 \\
    \end{align*}
    \text{Usual convergence condition: Bellman error over all states} < \Delta \\
    \text{Bellman error: } |v(s) - \min_{a} E\{c(s,a,s') + v(s')\}| \text{ for any } s \neq s_{\text{goal}}
\end{itemize}

\text{after backing up } s_{2
Computing Expected Cost Minimal Plans

\[ c(s_1, a_1, s_2) = 2 \]
\[ P(s_2|s_1, a_1) = 0.1 \]
\[ c(s_1, a_1, s_{goal}) = 2 \]
\[ P(s_{goal}|s_1, a_1) = 0.9 \]

### Value Iteration (VI):

Initialize \( v \)-values of all states to finite values;
Iterate over all \( s \) in MDP and re-compute until convergence:

\[ v(s_{goal}) = 0 \]
\[ v(s) = \min_a E\{ c(s, a, s') + v(s') \} \text{ for any } s \neq s_{goal} \]

**Usual convergence condition:** Bellman error over all states < \( \Delta \)

**Bellman error:**

\[ |v(s) - \min_a E\{ c(s, a, s') + v(s') \} | \text{ for any } s \neq s_{goal} \]

---

\( v = 4.41 \)
\( v = 2.41 \)
\( v = 5.41 \)
\( v = 4 \)
\( v = 1 \)

After backing up \( s_{start} \)
Computing Expected Cost Minimal Plans

• Value Iteration (VI):
  
  Initialize $v$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:

  $v(s_{goal}) = 0$

  $v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{goal}$

  Usual convergence condition: Bellman error over all states < $\Delta$

  Bellman error: $|v(s) - \min_a E\{c(s,a,s') + v(s')\}| \text{ for any } s \neq s_{goal}$
Computing Expected Cost Minimal Plans

• Value Iteration (VI):
  Initialize \( v \)-values of all states to finite values;
  Iterate over all \( s \) in MDP and re-compute until convergence:
  \[
  v(s_{goal}) = 0 \\
  v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{goal}
  \]
  Usual convergence condition: Bellman error over all states < \( \Delta \)
  Bellman error: \( |v(s) - \min_a E\{c(s,a,s') + v(s')\}| \) for any \( s \neq s_{goal} \)
Computing Expected Cost Minimal Plans

• Value Iteration (VI):
  Initialize $v$-values of all states to finite values;
  Iterate over all $s$ in MDP and re-compute until convergence:
  
  $v(s_{goal}) = 0$
  $v(s) = \min_a E\{c(s,a,s') + v(s')\}$ for any $s \neq s_{goal}$

  Usual convergence condition: Bellman error over all states $< \Delta$
  Bellman error: $|v(s) - \min_a E\{c(s,a,s') + v(s')\}|$ for any $s \neq s_{goal}$
Computing Expected Cost Minimal Plans

\[ v = 5.44444... \quad v = 4.44444... \]

\[ \text{Value Iteration (VI):} \]

Initialize \( v \)-values of all states to finite values;
Iterate over all \( s \) in MDP and re-compute until convergence:

\[ v(s_{\text{goal}}) = 0 \]
\[ v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}} \]

Usual convergence condition: Bellman error over all states < \( \Delta \)
Bellman error: \( |v(s) - \min_a E\{c(s,a,s') + v(s')\}| \text{ for any } s \neq s_{\text{goal}} \)
Computing Expected Cost Minimal Plans

Value Iteration (VI):
Initialize $v$-values of all states to finite values;
Iterate over all states in MDP and re-compute until convergence:

\[
\begin{align*}
    v^*(s) & = \min_a E\{c(s,a,s') + v(s')\} \\
    \text{for any } s \neq s_{\text{goal}}
\end{align*}
\]

Expected cost of executing greedy policy is at most:

\[
v^*(s_{\text{start}})c_{\text{min}} / (c_{\text{min}} - \Delta)
\]

where $c_{\text{min}}$ is minimum edge cost

Usual convergence condition: Bellman error over all states $< \Delta$

Bellman error: $|v(s) - \min_a E\{c(s,a,s') + v(s')\}|$ for any $s \neq s_{\text{goal}}$

At convergence...

optimal policy is given by greedy policy: always select an action that minimizes $E\{c(s,a,s') + v(s')\}$

every iteration computes one more decimal point
Computing Expected Cost Minimal Plans

- **Value Iteration (VI):**
  - Initialize \( v \)-values of all states to finite values.
  - Iterate over all states in MDP and re-compute until convergence:
    - \( v(s_{\text{goal}}) = 0 \)
    - \( v(s) = \min_a E\{c(s,a,s') + v(s')\} \) for any \( s \neq s_{\text{goal}} \)

  *Usual convergence condition: Bellman error over all states < \( \Delta \)*
  *Bellman error: \( |v(s) - \min_a E\{c(s,a,s') + v(s')\}| \) for any \( s \neq s_{\text{goal}} \)*

\[
\begin{align*}
S_1 & \quad \text{P}(s_{\text{goal}} | s_1, a_1) = 0.9 \\
S_2 & \quad c(s_1, a_1, s_2) = 2 \\
S_3 & \quad P(s_2 | s_1, a_1) = 0.1 \\
S_4 & \quad c(s_1, a_1, s_4) = 2 \\
S_5 & \quad P(s_4 | s_1, a_1) = 0.1
\end{align*}
\]

\[
\begin{align*}
v & = 4.44444... \quad v = 2.44444... \\
v & = 5.44444... \quad v = 1
\end{align*}
\]

\[
\begin{align*}
v & = 5.44444... \quad 1 \\
v & = 4 \quad 3 \\
v & = 1 \\
v & = 0
\end{align*}
\]

**Why condition?**
Computing Expected Cost Minimal Plans

Value Iteration (VI):

Initialize $v$-values of all states to finite values:

$v(s_{\text{goal}}) = 0$

Iterate over all $s$ in MDP and update:

$v(s) = \min_{a} E\{c(s,a,s') + v(s')\}$

Usual convergence condition: Bellman error over all states $< \Delta$

Bellman error: $|v(s) - \min_{a} E\{c(s,a,s') + v(s')\}|$ for any $s \neq s_{\text{goal}}$

How many backups required in a graph with no stochastic actions?
Computing Expected Cost Minimal Plans with RTDP

- **Real-time Dynamic Programming (RTDP)**
  - very popular alternative to Value Iteration
  - does **NOT** compute values of all states
  - focusses computations on states that are relevant
  - typically, **much more efficient than Value Iteration**
Computing Expected Cost Minimal Plans with RTDP

- **RTDP:**

  Initialize $v$-values of all states to admissible values;
  1. Follow greedy policy picking outcomes at random until goal is reached;
  2. Backup all states visited on the way;
  3. Reset to $s_{start}$ and repeat 1-3 until all states on the current greedy policy have Bellman errors $< \Delta$;
Computing Expected Cost Minimal Plans with RTDP

For any state $s$, picking action $a$ that minimizes $E\{c(s,a,s') + v(s')\}$

$\text{RTDP:}$

1. Initialize $v$-values of all states to admissible values;
2. Follow greedy policy picking outcomes at random until goal is reached;
3. Backup all states visited on the way;
4. Reset to $s_{start}$ and repeat 1-3 until all states on the current greedy policy have Bellman errors $< \Delta$;

$v = 5.44444... \ 1$
$v = 2.44444...$
$v = 4.44444...$
$v = 0$

$S_{start}$
$S_1$
$S_2$
$S_3$
$S_4$
$S_{goal}$

$P(s_{goal}|s_1,a_1) = 0.9$
$c(s_1,a_1,s_{goal}) = 2$

$P(s_2|s_1,a_1) = 0.1$
$c(s_1,a_1,s_2) = 2$

$P(s_3|s_1,a_1) = 0.1$
$c(s_1,a_1,s_3) = 2$

$P(s_4|s_1,a_1) = 0.1$
$c(s_1,a_1,s_4) = 2$
Computing Expected Cost Minimal Plans with RTDP

• RTDP:

  Initialize \( v \)-values of all states to admissible values;
  1. Follow greedy policy picking outcomes at random until goal is reached;
  2. Backup all states visited on the way;
  3. Reset to \( s_{\text{start}} \) and repeat 1-3 until all states on the current greedy policy have Bellman errors < \( \Delta \);

RTDP focuses its backups on what is relevant to the optimal plan rather than computing ALL state values (what VI does)
Computing Expected Cost Minimal Plans with RTDP

- **RTDP:**
  
  Initialize $v$-values of all states to admissible values;
  1. Follow greedy policy picking outcomes at random until goal is reached;
  2. Backup all states visited on the way;
  3. Reset to $s_{start}$ and repeat 1-3 until all states on the current greedy policy have Bellman errors $< \Delta$;

\[ c(s_1, a_1, s_{goal}) = 2 \]
\[ P(s_{goal}|s_1, a_1) = 0.9 \]
\[ P(s_2|s_1, a_1) = 0.1 \]

\[ v = 4.44444... \]
\[ v = 2.44444... \]
\[ v = 5.44444... \]
\[ v = 1 \]

RTDP converges in finite number of iterations (assuming goal is reachable from every state)
Computing Expected Cost Minimal Plans with RTDP

- RTDP:
  Initialize $v$-values of all states to admissible values;
  1. Follow greedy policy picking outcomes at random until goal is reached;
  2. Backup all states visited on the way;
  3. Reset to $s_{start}$ and repeat 1-3 until all states on the current greedy policy have Bellman errors $< \Delta$;

Expected cost of executing greedy policy is at most:
$$v*(s_{start})c_{min}/(c_{min}-\Delta)$$
where $c_{min}$ is minimum edge cost.
Rewards version of MDPs

• Suppose we have a Trash Collecting robot
  – its task is to go around the room and pick-up trash
  – if battery is dead, it can’t move anymore
  – available actions:
    • Look for trash (takes 1 min) and discovers trash with probability 0.4
    • Pick-up trash (takes 1 min), and receive reward of 100 units
    • Re-charge (takes 1 min). Battery level goes back to full 3 mins if successful with probability 0.9 (there is a chance that re-charge is not successful)
Markov Decision Processes, REWARDS version

• Optimal expected reward values $v^*$ satisfy:
  $$v^*(s) = \max_a E\{r(s,a,s') + \gamma v^*(s')\} \text{ for all } s$$
  (expectation over outcomes $s'$ of action $a$ executed at state $s$)

• Optimal policy $\pi^*$:
  $$\pi^*(s) = \arg\max_a E\{r(s,a,s') + \gamma v^*(s')\}$$

• Computing optimal $v^*$-values via value iteration (VI):
  re-compute $v(s) = \max_a E\{r(s,a,s') + \gamma v(s')\}$ until convergence
Markov Decision Processes, REWARDS version

- Optimal expected reward values $v^*$ satisfy:
  \[ v^*(s) = \max_a E\{r(s,a,s') + \gamma v^*(s')\} \text{ for all } s \]
  (expectation over outcomes $s'$ of action $a$ executed at state $s$)

- Optimal policy $\pi^*$:
  \[ \pi^*(s) = \arg\max_a E\{r(s,a,s') + \gamma v^*(s')\} \]

- Computing optimal $v^*$-values via value iteration (VI):
  re-compute $v(s) = \max_a E\{r(s,a,s') + \gamma v(s')\}$ until convergence

- discount factor in $(0, 1]$ (e.g., 0.95) especially useful when there is no goal
What You Should Know…

• Pros and Cons of solving Expected Cost formulation (rather than Minimax formulation)

• The operation of Value Iteration

• The operation of RTDP

• Rewards formulation of MDPs and when it should be used