15-466
Computer Game Programming

Intelligence II: Advanced Decision-Making Mechanisms

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Advanced Decision-making Mechanisms for this Class

• More on Behavior Trees

• Planning to Achieve the Goal

• Planning with Uncertainty to Achieve the Goal
Advanced Decision-making Mechanisms for this Class

• More on Behavior Trees

• Planning to Achieve the Goal

• Planning with Uncertainty to Achieve the Goal
More on Behavior Trees

• Additional task: *Decorator*
• Has only one child whose execution it controls in a special way

*Example:*

“*Filter decorators*” decide whether to execute its children based on some conditions.
More on Behavior Trees

- Additional task: *Decorator*
- Has only one child whose execution it controls in a special way

*Example:*

“Loop decorators” loop until failure

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
More on Behavior Trees

• Additional task: *Decorator*
• Has only one child whose execution it controls in a special way

Example:
“Loop decorators” loop until fail

What about behavior trees for groups of characters?

---

from “Artificial Intelligence for Games” by I. Millington & J. Funge
More on Behavior Trees

• Additional task: *Parallel*
• Executes tasks in parallel until first fails or all succeed

*Example:*

*from “Artificial Intelligence for Games” by I. Millington & J. Funge*
More on Behavior Trees

- Additional task: Parallel
- Executes tasks in parallel until first fails or all succeed

More complex example:

```
Interrupter
  └── Keep robbing the bank?
    └── Until fail
        └── Police arrived?

What does it do?
```
Advanced Decision-making Mechanisms for this Class

• More on Behavior Trees

• Planning to Achieve the Goal

• Planning with Uncertainty to Achieve the Goal
Goal-Oriented Behavior

• Beyond hard-coding stimulus-response pairs
• Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)
Goal-Oriented Behavior

• Beyond hard-coding stimulus-response pairs
• Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Three components:

Goals (motives),
How Pressing each Goal is (insistence)
Actions with Expected Impact on the Insistence of Each Goal
Goal-Oriented Behavior

• Beyond hard-coding stimulus-response pairs
• Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Example:

Goals with Insistence Values:

Eat = 9, Kill Enemy = 8, Get Healthy = 4

Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5)
Kill Enemy (Kill Enemy: -8, Get Healthy: +4)
Get Health Pack (Get Healthy: –2)
Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

*Example:*

**Goals with Insistence Values:**

Eat = 9, Kill Enemy = 8, Get Healthy = 4

**Actions with Impact on Insistence Values of Goals:**

Get Food (Eat: -5)

Kill Enemy (Kill Enemy: -8, Get Healthy: +4)

Get Health Pack (Get Healthy: −2)

Pick action that has the best net effect (could be weighted)
Goal-Oriented Behavior

• Beyond hard-coding stimulus-response pairs
• Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Example:

Goals with Insistence Values:

\[ \text{Eat} = 9, \text{Kill Enemy} = 8, \text{Get Healthy} = 4 \]

Actions with Impact on Insistence Values of Goals:

Get Food (Eat: -5)
Kill Enemy (Kill Enemy: -8, Get Healthy: +4)
Get Health Pack (Get Healthy: –2)

Potential problems?

Pick action that has the best net effect (could be weighted)

Negative side effect of the action
Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Example:
Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies.

Goals with Insistence Values:
\[ \text{Eat} = 7, \text{Kill Enemy} = 8, \text{Get Healthy} = 4, \text{NewWeapons} = 1 \]

Actions with Impact on Insistence Values of Goals:
- Get Food (Eat: -5, Get Healthy: +2)
- Kill Enemy (Kill Enemy: -8, Get Healthy: +7)
- Get Health Pack (Get Healthy: -2, Eat: +2)
- Get Weapon (NewWeapons: -1, Get Healthy: +8
  plus Kill Enemy action will have no impact on Health)
Goal-Oriented Behavior

- Beyond hard-coding stimulus-response pairs
- Seeks to satisfy internal goals (e.g., hunger, threat, gold, etc.)

Example:
Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies.

Goals with Insistence Values:

\[ \text{Eat} = 7, \text{Kill Enemy} = 8, \text{Get Healthy} = 4, \text{New Weapons} = 1 \]

Actions with Impact on Insistence Values of Goals:

- Get Food (Eat: -5, Get Healthy: +2)
- Kill Enemy (Kill Enemy: -8, Get Healthy: +7)
- Get Health Pack (Get Healthy: -2, Eat: +2)
- Get Weapon (New Weapons: -1, Get Healthy: +8 plus Kill Enemy action will have no impact on Health)
Goal-Oriented Planning

• Beyond single-step decisions

Search-space (graph) for finding a goal that satisfies the necessary constraints:

Example:

initial state is 0000
goal state is 1111
(e.g., bit 0: food is picked up,
bit 1: health pack is picked up,
bit 2: weapon is picked up,
bit 3: enemy is shot)
Goal-Oriented Planning

• Beyond single-step decisions

Search-space (graph) for finding a goal that satisfies the necessary constraints:

The validity of a transition depends on the source state (e.g., can’t shoot enemy if no weapon was picked up)

Example:

initial state is 0000
goal state is 1111
(e.g., bit 0: food is picked up,
bit 1: health pack is picked up,
bit 2: weapon is picked up,
bit 3: enemy is shot)
Goal-Oriented Planning

• Beyond single-step decisions

Search-space (graph) for finding a goal that satisfies the necessary constraints:

Example:
initial state is 0000
goal state is 1111
(e.g., bit 0: food is picked up,
bit 1: health pack is picked up,
bit 2: weapon is picked up,
bit 3: enemy is shot)
Goal-Oriented Planning

• Beyond single-step decisions

Search-space (graph) for finding a partially-defined goal:

Example:

initial state is 0000
goal state is 1111
(e.g., bit 0: food is picked up,
bit 1: health pack is picked up,
bit 2: weapon is picked up,
bit 3: enemy is shot)

What if the goal is to shoot enemy (whether health pack and food are picked up or not)?
Goal-Oriented Planning

- Beyond single-step decisions

Search-space (graph) for finding a partially-defined goal:

Example:
- initial state is 0000
- goal state is 1111
  (e.g., bit 0: food is picked up,
   bit 1: health pack is picked up,
   bit 2: weapon is picked up,
   bit 3: enemy is shot)

What are the efficient way to represent the state vectors?
Goal-Oriented Planning

• Beyond single-step decisions

\[ S_1 \]

- \[ \text{Hunger: 7; Enemy shot: 0; Health: 4; Weapon: 1} \]

Working out Example with some non-binary variables:

Goals with Insistence Values:

\[ \text{Eat} = 7, \text{Kill Enemy} = 8, \text{Get Healthy} = 4, \text{NewWeapons} = 1 \]

Actions with Impact on Insistence Values of Goals:

- Get Food (\[ \text{Eat: -5, Get Healthy: +2} \])
- Kill Enemy (\[ \text{Kill Enemy: -8, Get Healthy: +7} \])
- Get Health Pack (\[ \text{Get Healthy: -2, Eat: +2} \])
- Get Weapon (\[ \text{NewWeapons: -1, Get Healthy: +8} \]
  plus Kill Enemy action will have no impact on Health)
Goal-Oriented Planning

- Beyond single-step decisions

\[
S_1 \quad \text{Hunger: 7; Enemy shot: 0; Health: 4; Weapon: 1}
\]

Construct the rest of the graph and find optimal plan

Working out Example with some non-binary variables:

**Goals with Insistence Values:**

\[
\text{Eat} = 7, \text{Kill Enemy} = 8, \text{Get Healthy} = 4, \text{NewWeapons} = 1
\]

**Actions with Impact on Insistence Values of Goals:**

- **Get Food** (Eat: -5, Get Healthy: +2)
- **Kill Enemy** (Kill Enemy: -8, Get Healthy: +7)
- **Get Health Pack** (Get Healthy: -2, Eat: +2)
- **Get Weapon** (NewWeapons: -1, Get Healthy: +8 plus Kill Enemy action will have no impact on Health)
Goal-Oriented Planning

- Beyond single-step decisions

**Search-space (graph) for finding a partially-defined goal:**

**Example:**
- initial state is 0000
- goal state is 1111
  - (e.g., bit 0: food is picked up,
    bit 1: health pack is picked up,
    bit 2: weapon is picked up,
    bit 3: enemy is shot)
Goal-Oriented Planning

- Beyond single-step decisions

Search-space (graph) for finding a partially-defined goal:

Example:
- initial state is 0000
- goal state is 1111
  (e.g., bit 0: food is picked up,
   bit 1: health pack is picked up,
   bit 2: weapon is picked up,
   bit 3: enemy is shot)

How to search the graph?
DFS, BFS, A*, etc.
Goal-Oriented Planning

• Beyond single-step decisions

Search-space (graph) for partially defined goal:

Example:
  initial state is 0000
  goal state is 1111
(e.g., bit 0: food is picked up,
bit 1: health pack is picked up,
bit 2: weapon is picked up,
bit 3: enemy is shot)

How to search the graph?

DFS, BFS, A*, etc.

Problems with BFS and A* for large state vectors?
Goal-Oriented Planning

- Beyond single-step decisions
- **IDA**: Very popular search for state-spaces with large branching factors and shallow goals
Goal-Oriented Planning

- Beyond single-step decisions
- **IDA**: Very popular search for state-spaces with large branching factors and shallow goals

**IDA** (Iterative Deepening A*)

1. set $f_{\text{max}} = 1$ (or some other small value)
2. traverse the graph in DFS fashion without expanding states with $f > f_{\text{max}}$
3. If path to the goal found, return the best path it finds
4. Otherwise $f_{\text{max}} = f_{\text{max}} + 1$ and go to step 2
Goal-Oriented Planning

• Beyond single-step decisions
• IDA*: Very popular search for state-spaces with large branching factors and shallow goals

IDA* (Iterative Deepening A*)

1. set $f_{\text{max}} = 1$ (or some other small value)
2. traverse the graph in DFS fashion without expanding states with $f>f_{\text{max}}$
3. If path to the goal found, return the best path it finds
4. Otherwise $f_{\text{max}} = f_{\text{max}} + 1$ and go to step 2

Proof?

• Complete and optimal in any state-space (with positive costs)

• Memory: $O(bl)$, where $b$ – max. branching factor, $l$ – length of optimal path

• Complexity: $O(kbl)$, where $k$ is the number of times DFS is called
Advanced Decision-making Mechanisms for this Class

- More on Behavior Trees

- Planning to Achieve the Goal

- Planning with Uncertainty to Achieve the Goal
Goal-Oriented Planning Under Uncertainty

• Dealing with uncertainty in outcomes

Example:
Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies.
Suppose also there is 50% chance of getting health pack at each attempt.

Goals with Insistence Values:
Eat = 7, Kill Enemy = 8, Get Healthy = 4, New Weapons = 1

Actions with Impact on Insistence Values of Goals:
Get Food (Eat: -5, Get Healthy: +2)
Kill Enemy (Kill Enemy: -8, Get Healthy: +7)
Get Health Pack (Get Healthy: -2, Eat: +2)
Get Weapon (New Weapons: -1, Get Healthy: +8 plus Kill Enemy action will have no impact on Health)
Goal-Oriented Planning Under Uncertainty

• Dealing with uncertainty in outcomes

Example:
Suppose the character is under attack and can pick a weapon that allows it to more effectively shoot the enemies.
Suppose also there is 50% chance of getting health pack at each attempt.

- S1
  - Action: get health pack
  - P=50%
  - S2: Hunger: 9; Enemy shot: 0; Health: 2; Weapon: 1
  - P=50%
  - S3: Hunger: 9; Enemy shot: 0; Health: 4; Weapon: 1
Planning in MDPs

- What plan to compute?
  - Plan that minimizes the worst-case scenario (minimax plan)
  - Plan that minimizes the expected cost
  - Plan that minimizes cost while guaranteeing $P(\text{goal reached}) > t$

- Without uncertainty, plan is a single path:
  a sequence of states (a sequence of actions)

- In MDPs, plan is a policy $\pi$:
  mapping from a state onto an action
Planning in MDPs

- What plan to compute?
  - Plan that minimizes the worst-case scenario (minimax plan)
  - Plan that minimizes the expected cost
  - Plan that minimizes cost while guaranteeing $P(\text{goal reached}) > t$

- Without uncertainty, plan is a single path:
  a sequence of states (a sequence of actions)

- In MDPs, plan is a policy $\pi$:
  mapping from a state onto an action

Which ones are policies?

Why?
Minimax Formulation

- Optimal policy $\pi^*$:
  minimizes the worst cost-to-goal
  $\pi^* = \arg\min_\pi \max_{\text{outcomes of } \pi} \{\text{cost-to-goal}\}$

- worst cost-to-goal for $\pi_1 = (s_{\text{start}}, s_2, s_4, s_3, s_{\text{goal}})$ is:
  $1+1+3+1 = 6$

- worst cost-to-goal for $\pi_2 = (\text{try to go through } s_1)$ is:
  $1+2+2+2+2+2+2+\ldots = \infty$
Minimax Formulation

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

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

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

\[ P(s_{\text{goal}} | s_1, a_1) = 0.1 \]

• Optimal policy \( \pi^* \):
  minimizes the worst cost-to-goal

\[ \pi^* = \arg \min_{\pi} \max \text{ outcomes of } \pi \{ \text{cost-to-goal} \} \]

• Optimal minimax policy \( \pi^* = \pi_1 = (s_{\text{start}}, s_2, s_4, s_3, s_{\text{goal}}) \)

What are potential problems with minimax approaches?
Expected Cost Formulation

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

\[ \pi^* = \arg\min_\pi E\{\text{cost-to-goal}\} \]

- expected cost-to-goal for $\pi_1 = (s_{\text{start}}, s_2, s_4, s_3, s_{\text{goal}})$ is

\[ 1+1+3+1=6 \]

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

\[ 0.9*(1+2+2) + 0.9*0.1*(1+2+2+2+2) + 0.9*0.1*0.1*(1+2+2+2+2+2+2+2) + \ldots = 5.444 \]
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 $\pi_1 = (s_{\text{start}}, s_2, s_4, s_3, s_{\text{goal}})$ is
  $1 + 1 + 3 + 1 = 6$

- cost-to-goal for $\pi_2 = (\text{try to go through } s_1)$ 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
  $\pi^* = \operatorname{argmin}_{\pi} E\{\text{cost-to-goal}\}$

• Optimal expected cost policy $\pi^* = \pi_2 = (\text{go through } s_1)$
Minimizing Cost with $P(\text{success}) > t$ constraint

• Optimal path $\pi^*$ is a path that minimizes the cost-to-goal assuming some outcomes
  $\pi^* = \operatorname{argmin}_\pi \{\text{cost-to-goal}\}$
  s.t. $P(\text{assumed outcomes}) > t$

• for $\pi_1 = (s_{\text{start}}, s_2, s_4, s_3, s_{\text{goal}})$:
  cost-to-goal = $1+1+3+1 = 6$, $P(\text{success}) = 1$

• for $\pi_2 = (\text{try to go through } s_1)$ is:
  cost-to-goal = $1+2+2 = 5$, $P(\text{success}) = 0.9$
Minimizing Cost with $P(\text{success}) > t$ constraint

- Optimal path $\pi^*$ is a path that minimizes the cost-to-goal assuming some outcomes $\pi^* = \arg\min_{\pi} \{\text{cost-to-goal}\}$ s.t. $P(\text{assumed outcomes}) > t$

  - for $t > 0.9$, $\pi^* = \pi_1 = (s_{\text{start}}, s_2, s_4, s_3, s_{\text{goal}})$
  
  - for $t \leq 0.9$, $\pi^* = \pi_2 = (\text{try to go through } s_1)$
Computing Expected Cost Minimal Plans

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

- Let $v^*(s)$ 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 \( s \))

\[
\begin{align*}
S_{\text{start}} & \rightarrow S_2 & S_2 & \rightarrow S_1 & a_1 & \rightarrow S_{\text{goal}} \\
1 & & 2 & & P(s_{\text{goal}}|s_1,a_1)=0.9 & c(s_1,a_1,s_{\text{goal}}) = 2 \\
S_4 & \rightarrow S_2 & S_1 & \rightarrow S_{\text{goal}} \\
1 & & 1 & & c(s_1,a_1,s_2) = 2 & P(s_{\text{goal}}|s_1,a_1)=0.1 \\
S_4 & \rightarrow S_3 & S_3 \\
3 & & 1 & & \\
\end{align*}
\]
Computing Expected Cost Minimal Plans

- Optimal expected cost-to-goal values $v^*$ satisfy:
  \[
  v^*(s_{goal}) = 0 \\
  v^*(s) = \min_a E\{c(s,a,s') + v^*(s')\} \text{ for all } s \neq s_{goal} \\
  \text{(expectation over outcomes } s' \text{ of action } a \text{ executed at state } s) \\
  \]

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

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

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

  *convergence time does depend a lot on iteration 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:

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

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

convergence time does depend a lot on iteration order

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

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

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

![Diagram of a Markov Decision Process (MDP)]

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

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

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

---

- Diagram:
  - States: $S_{start}$, $S_1$, $S_2$, $S_3$, $S_4$, $S_{goal}$
  - Transitions: $P(s_{goal}|s_1,a_1) = 0.9$,
    $c(s_1,a_1,s_{goal}) = 2$
  - Values:
    - $v(S_2) = 1$
    - $v(S_1) = 2$
    - $v(S_3) = 1$
    - $v(S_4) = 0$
    - $v(S_{goal}) = 0$ after backing up $S_3$
Computing Expected Cost Minimal Plans

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

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

\[ P(s_{\text{goal}}|s_1, a_1) = 0.1 \]

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

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

- 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:
  
  $$v(s_{goal}) = 0$$
  $$v(s) = \min_{a} E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{goal}$$

  *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

\begin{itemize}
\item Value Iteration (VI):
\end{itemize}

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

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

- 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:
  
  $v(s_{goal}) = 0$
  
  $v(s) = \min_a E\{c(s,a,s') + v(s')\}$ 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 $s \neq s_{goal}$

How to select backups more effectively?
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:
  \[
  v(s_{goal}) = 0 \\
  v(s) = \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{goal}
  \]

  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

\[
\begin{align*}
S_4 & \xrightarrow{3} S_3 & v=4 \\ S_3 & \xrightarrow{1} a_1 & c(s_1,a_1,s_2) = 2 \\ S_1 & \xleftarrow{1} P(s_{goal}|s_1,a_1)=0.9 \\
& & P(s_{goal}|s_1,a_1)=0.1 \\ S_2 & \xrightarrow{2} v=4.41 \\ S_1 & \xrightarrow{v=2.41} \text{(after backing up } s_2) \\
S_{start} & \xrightarrow{v=5.1} 1
\end{align*}
\]

- **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

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

\begin{align*}
  c(s_1, a_1, s_2) &= 2 \\
  P(s_{goal}|s_1, a_1) &= 0.9 \\
  c(s_1, a_1, s_{goal}) &= 2
\end{align*}

\begin{align*}
  P(s_{goal}|s_1, a_1) &= 0.1 \\
  v(s) &= \min_a E\{c(s, a, s') + v(s')\} \text{ for any } s \neq s_{goal}
\end{align*}

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

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

\[ c(s_1, a_1, s_{\text{goal}}) = 2 \]
\[ c(s_1, a_1, s_2) = 2 \]
\[ P(s_{\text{goal}}|s_1, a_1) = 0.9 \]
\[ P(s_2|s_1, a_1) = 0.1 \]
\[ P(s_3|s_2) = 0.1 \]
\[ P(s_4|s_2) = 0.3 \]
\[ v(s_1) = 2.441 \]
\[ v(s_2) = 4.441 \]
\[ v(s_3) = 1 \]
\[ v(s_4) = 5.41 \]

• 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 \( s \) in MDP and re-compute until convergence:

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

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 \( s \) in MDP and re-compute until convergence:

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

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

At convergence...

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 $s$ in MDP and re-compute until convergence:

$$\begin{align*}
v(s_{\text{goal}}) &= 0 \\
v(s) &= \min_a E\{c(s,a,s') + v(s')\} \text{ for any } s \neq s_{\text{goal}}
\end{align*}$$

At convergence...

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

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

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

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

  $\nu(s_{goal}) = 0$

  $\nu(s) = \min_a E\{c(s,a,s') + \nu(s')\}$ 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 $s \neq s_{goal}$

VI converges in finite number of iterations (assuming goal is reachable from every state)

Why condition?
• 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 $s \neq s_{goal}$*