CS 747, Autumn 2023: Lecture 11

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2023
Markov Decision Problems

1. Policy iteration: variants and complexity bounds

2. Analysis of bounds
   - Basic tools
   - Howard’s PI with $k = 2$
   - BSPI with $k = 2$
   - Open problems

3. Review of MDP planning
Markov Decision Problems

1. Policy iteration: variants and complexity bounds

2. Analysis of bounds
   - Basic tools
   - Howard’s PI with $k = 2$
   - BSPI with $k = 2$
   - Open problems

3. Review of MDP planning
Policy Iteration Algorithm

\[ \pi \leftarrow \text{Arbitrary policy.} \]

**While** \( \pi \) has improvable states:

\[ \pi' \leftarrow \text{PolicyImprovement}(\pi). \]

\[ \pi \leftarrow \pi'. \]

**Return** \( \pi \).
Policy Iteration Algorithm

\[ \pi \leftarrow \text{Arbitrary policy.} \]

\textbf{While} \( \pi \) has improvable states:

\[ \pi' \leftarrow \text{PolicyImprovement}(\pi). \]

\[ \pi \leftarrow \pi'. \]

\textbf{Return} \( \pi \).
Policy Iteration Algorithm

\( \pi \leftarrow \text{Arbitrary policy.} \)

**While** \( \pi \) has improvable states:

\( \pi' \leftarrow \text{PolicyImprovement}(\pi) \).

\( \pi \leftarrow \pi' \).

**Return** \( \pi \).
Policy Iteration Algorithm

$$\pi \leftarrow \text{Arbitrary policy.}$$

**While** $$\pi$$ has improvable states:

$$\pi' \leftarrow \text{PolicyImprovement}(\pi).$$

$$\pi \leftarrow \pi'.$$

**Return** $$\pi.$$
Policy Iteration Algorithm

\[ \pi \leftarrow \text{Arbitrary policy.} \]

**While** \( \pi \) has improvable states:

\[ \pi' \leftarrow \text{PolicyImprovement}(\pi). \]

\[ \pi \leftarrow \pi'. \]

**Return** \( \pi. \)
Policy Iteration Algorithm

\[ \pi \leftarrow \text{Arbitrary policy.} \]

**While** \( \pi \) has improvable states:

\[ \pi' \leftarrow \text{PolicyImprovement}(\pi). \]

\[ \pi \leftarrow \pi'. \]

**Return** \( \pi. \)
Policy Iteration Algorithm

\[ \pi \leftarrow \text{Arbitrary policy.} \]

**While** \( \pi \) has improvable states:

\[ \pi' \leftarrow \text{PolicyImprovement}(\pi). \]

\[ \pi \leftarrow \pi'. \]

**Return** \( \pi \).
Policy Iteration Algorithm

\[ \pi \leftarrow \text{Arbitrary policy.} \]

**While** \( \pi \) has improvable states:

\[ \pi' \leftarrow \text{PolicyImprovement}(\pi). \]

\[ \pi \leftarrow \pi'. \]

**Return** \( \pi \).

Path taken (and hence the number of iterations) in general depends on the **switching strategy**.
Howard’s Policy Iteration

- Greedy; switch all improvable states.
Random Policy Iteration

- Switch a non-empty subset of improvable states chosen uniformly at random.
Random Policy Iteration

- Switch a non-empty subset of improvable states chosen uniformly at random.
Random Policy Iteration

- Switch a non-empty subset of improvable states chosen uniformly at random.
Simple Policy Iteration

- Assume a fixed indexing of states.
- Switch the improvable state with the highest index.
Upper and Lower Bounds

$U(n, k)$ is an upper bound applicable to a set of PI variants $\mathcal{L}$ if

- for each $n$-state, $k$-action MDP $M = (S, A, T, R, \gamma)$,
- for each policy $\pi : S \rightarrow A$,
- for each algorithm $L \in \mathcal{L}$,

the expected number of policy evaluations performed by $L$ on $M$ if initialised at $\pi$ is at most $U(n, k)$. 
**Upper and Lower Bounds**

$U(n, k)$ is an upper bound applicable to a set of PI variants $\mathcal{L}$ if

- for each $n$-state, $k$-action MDP $M = (S, A, T, R, \gamma)$,
- for each policy $\pi : S \rightarrow A$,
- for each algorithm $L \in \mathcal{L}$,
the expected number of policy evaluations performed by $L$ on $M$ if initialised at $\pi$ is at most $U(n, k)$.

$X(n, k)$ is a lower bound applicable to a set of PI variants $\mathcal{L}$ if

- there exists an $n$-state, $k$-action MDP $M = (S, A, T, R, \gamma)$,
- there exists a policy $\pi : S \rightarrow A$,
- there exists an algorithm $L \in \mathcal{L}$,
such that the expected number of policy evaluations performed by $L$ on $M$ if initialised at $\pi$ is at least $X(n, k)$.
# Switching Strategies and Bounds

## Upper bounds on number of iterations

<table>
<thead>
<tr>
<th>PI Variant</th>
<th>Type</th>
<th>$k = 2$</th>
<th>General $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard’s (Greedy) PI [H60, MS99]</td>
<td>Deterministic</td>
<td>$O\left(\frac{2^n}{n}\right)$</td>
<td>$O\left(\frac{k^n}{n}\right)$</td>
</tr>
<tr>
<td>Mansour and Singh’s Random PI [MS99]</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
<td>$O\left(\frac{k^n}{2}\right)$</td>
</tr>
<tr>
<td>Mansour and Singh’s Random PI [HPZ14]</td>
<td>Randomised</td>
<td>$\text{poly}(n) \cdot 1.5^n$</td>
<td>–</td>
</tr>
</tbody>
</table>
Switching Strategies and Bounds

### Upper bounds on number of iterations

<table>
<thead>
<tr>
<th>PI Variant</th>
<th>Type</th>
<th>General $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard’s (Greedy) PI [H60, MS99]</td>
<td>Deterministic</td>
<td>$O\left(\frac{2^n}{n}\right)$</td>
</tr>
<tr>
<td>Mansour and Singh’s Random PI [MS99]</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
</tr>
<tr>
<td>Mansour and Singh’s Random PI [HPZ14]</td>
<td>Randomised</td>
<td>$\text{poly}(n) \cdot 1.5^n$</td>
</tr>
</tbody>
</table>

### Lower bounds on number of iterations

$\Omega(n)$: Howard’s PI on $n$-state, 2-action MDPs [HZ10].
## Switching Strategies and Bounds

### Upper bounds on number of iterations

<table>
<thead>
<tr>
<th>PI Variant</th>
<th>Type</th>
<th>Type</th>
<th>General $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard’s (Greedy) PI [H60, MS99]</td>
<td>Deterministic</td>
<td>$O\left(\frac{2^n}{n}\right)$</td>
<td>$O\left(\frac{k^n}{n}\right)$</td>
</tr>
<tr>
<td>Mansour and Singh’s Random PI [MS99]</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
<td>$\approx O\left(\frac{k^n}{2}\right)$</td>
</tr>
<tr>
<td>Mansour and Singh’s Random PI [HPZ14]</td>
<td>Randomised</td>
<td>poly$(n) \cdot 1.5^n$</td>
<td>–</td>
</tr>
</tbody>
</table>

### Lower bounds on number of iterations

- $\Omega(n)$ Howard’s PI on $n$-state, 2-action MDPs [HZ10].
- $\Omega(2^n)$ Simple PI on $n$-state, 2-action MDPs [MC94].
PI: Some Recent Results

(Polynomial factors ignored. Authors with names underlined once took CS 747!)
Pl: Some Recent Results

(Polynomial factors ignored. Authors with names underlined once took CS 747!)

- Kalyanakrishnan, Mall, and Goyal (2016) devise the Batch-switching PI algorithm (deterministic), and show an upper bound of $1.6479^n$ for $k = 2$. 
Kalyanakrishnan, Mall, and Goyal (2016) devise the Batch-switching PI algorithm (deterministic), and show an upper bound of $1.6479^n$ for $k = 2$.

Gupta and Kalyanakrishnan (2017) give a deterministic PI variant with upper bound $k^{0.7207n}$. Taraviya and Kalyanakrishnan (2019) improve to $k^{0.7019n}$. 

Ashutosh, Consul, Dedhia, Khirwadkar, Shah, and Kalyanakrishnan (2020) show a lower bound of $\sqrt{k}n$ iterations for a deterministic variant of PI.
Pi: Some Recent Results

(Polynomial factors ignored. Authors with names underlined once took CS 747!)

- Kalyanakrishnan, Mall, and Goyal (2016) devise the Batch-switching PI algorithm (deterministic), and show an upper bound of $1.6479^n$ for $k = 2$.
- Gupta and Kalyanakrishnan (2017) give a deterministic PI variant with upper bound $k^{0.7207n}$. Taraviya and Kalyanakrishnan (2019) improve to $k^{0.7019n}$.
- Kalyanakrishnan, Misra, and Gopalan (2016) show an upper bound of $(2 + \ln(k - 1))^n$ for a randomised PI variant.
PI: Some Recent Results

(Polynomial factors ignored. Authors with names underlined once took CS 747!)

- Kalyanakrishnan, Mall, and Goyal (2016) devise the Batch-switching PI algorithm (deterministic), and show an upper bound of \(1.6479^n\) for \(k = 2\).
- Gupta and Kalyanakrishnan (2017) give a deterministic PI variant with upper bound \(k^{0.7207n}\). Taraviya and Kalyanakrishnan (2019) improve to \(k^{0.7019n}\).

- Kalyanakrishnan, Misra, and Gopalan (2016) show an upper bound of \((2 + \ln(k - 1))^n\) for a randomised PI variant.
- Taraviya and Kalyanakrishnan (2019) show an upper bound of \((O(\sqrt{k \log(k)}))^n\) for a randomised variant of Howard’s PI.
Pi: Some Recent Results

(Polynomial factors ignored. Authors with names underlined once took CS 747!)

- Kalyanakrishnan, Mall, and Goyal (2016) devise the Batch-switching PI algorithm (deterministic), and show an upper bound of \(1.6479^n\) for \(k = 2\).
- Gupta and Kalyanakrishnan (2017) give a deterministic PI variant with upper bound \(k^{0.7207n}\). Taraviya and Kalyanakrishnan (2019) improve to \(k^{0.7019n}\).

- Kalyanakrishnan, Misra, and Gopalan (2016) show an upper bound of \((2 + \ln(k - 1))^n\) for a randomised PI variant.
- Taraviya and Kalyanakrishnan (2019) show an upper bound of \((O(\sqrt{k \log(k)}))^n\) for a randomised variant of Howard’s PI.

- Ashutosh, Consul, Dedhia, Khirwadkar, Shah, and Kalyanakrishnan (2020) show a lower bound of \(\sqrt{k}^n\) iterations for a deterministic variant of PI.
Markov Decision Problems

1. Policy iteration: variants and complexity bounds

2. Analysis of bounds
   - Basic tools
   - Howard’s PI with $k = 2$
   - BSPI with $k = 2$
   - Open problems

3. Review of MDP planning
1. Policy Improvement and Policy “Deprovement”

\[ \pi' \succeq \pi. \]

Policy Improvement

Shivaram Kalyanakrishnan (2023)
1. Policy Improvement and Policy “Deprovement”

\[ \pi' \succ \pi \]

\[ \pi \geq \pi'' \]
2. Improvement sets in 2-action MDPs

Non-optimal policies $\pi, \pi' \in \Pi$ cannot have the same set of improvable states.
2. Improvement sets in 2-action MDPs

Non-optimal policies $\pi, \pi' \in \Pi$ cannot have the same set of improvable states.

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1
2. Improvement sets in 2-action MDPs

Non-optimal policies $\pi, \pi' \in \Pi$ cannot have the same set of improvable states.

1 1 0 0 0 1 1 0

$\succ$

1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1
Non-optimal policies $\pi, \pi' \in \Pi$ cannot have the same set of improvable states.

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
\succ
\end{array}
\]

\[
\begin{array}{ccccccccc}
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
\succeq
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]
2. Improvement sets in 2-action MDPs

Non-optimal policies $\pi, \pi' \in \Pi$ cannot have the same set of improvable states.

\[
\begin{array}{cccccccccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
\checkmark & \checkmark \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
\checkmark \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\]
2. Improvement sets in 2-action MDPs

Non-optimal policies $\pi, \pi' \in \Pi$ cannot have the same set of improvable states.

\[
\begin{array}{cccccccccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
\triangleright & & & & & & & & & & & & & \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
\triangleright & \triangleright & & & & & & & & & & & & & \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}
\]

Contradiction!
2. Improvement sets in 2-action MDPs

Non-optimal policies $\pi, \pi' \in \Pi$ cannot have the same set of improvable states.

Contradiction!
Markov Decision Problems

1. Policy iteration: variants and complexity bounds

2. Analysis of bounds
   - Basic tools
   - Howard’s PI with $k = 2$
   - BSPI with $k = 2$
   - Open problems

3. Review of MDP planning
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

\[ \pi \]

\[
\begin{array}{cccccccc}
\pi_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\pi_1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\pi_2 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\pi_3 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\pi_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
\]
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

\[
\pi' \quad 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1
\]

\[
\pi \quad 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0
\]
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

Possible?

\[
\begin{array}{ccccccc}
\pi' & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

\[ \pi' = \begin{array}{ccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array} \]

\[ \pi = \begin{array}{ccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \]
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

\[
\begin{align*}
\pi' & : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \\
\pi_1 & : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \\
\pi & : 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0
\end{align*}
\]
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

\[
\begin{align*}
\pi' & : 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \\
\pi_1 & : 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \\
\pi_2 & : 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \\
\pi & : 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0
\end{align*}
\]
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

<table>
<thead>
<tr>
<th>$\pi'$</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\pi_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\pi$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Howard’s Policy Iteration (2-action MDPs)

Switch actions in **every** improvable state.

<table>
<thead>
<tr>
<th></th>
<th>$\pi'$</th>
<th>$\pi_1$</th>
<th>$\pi_2$</th>
<th>$\pi_3$</th>
<th>$\pi_4$</th>
<th>$\pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

If $\pi$ has $m$ improvable states and $\pi \xrightarrow{Howard’s PI} \pi'$, then there exist $m$ policies $\pi''$ such that $\pi' \succeq \pi'' \succ \pi$. 

Shivaram Kalyanakrishnan (2023)
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

If $\pi$ has $m$ improvable states and $\pi \xrightarrow{\text{Howard's PI}} \pi'$, then there exist $m$ policies $\pi''$ such that $\pi' \succeq \pi'' \succ \pi$. 

$$
\begin{align*}
\pi' &= 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \\
\pi_1 &= 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \\
\pi_2 &= 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \\
\pi_3 &= 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \\
\pi_4 &= 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \\
\pi &= 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 
\end{align*}
$$
Howard’s Policy Iteration (2-action MDPs)

- Take $m^* = \frac{n}{3}$. 

Number of policies with $m^* \text{ or more improvable states visited} \leq 2^n m^* = 2^n \frac{n}{3}$.

Number of policies with fewer than $m^* \text{ improvable states visited} \leq n^0 + n^1 + n^2 + \cdots + n^{m^* - 1} \leq 32^n n^n$.

Number of iterations taken by Howard’s PI: $O(2^n n^n)$ [MS99, HGDJ14].
Howard’s Policy Iteration (2-action MDPs)

- Take $m^* = \frac{n}{3}$.
- Number of policies with $m^*$ or more improvable states visited
Howard’s Policy Iteration (2-action MDPs)

- Take $m^* = \frac{n}{3}$.
- Number of policies with $m^*$ or more improvable states visited

$$\leq \frac{2^n}{m^*} = \frac{2^n}{n/3}.$$
Howard’s Policy Iteration (2-action MDPs)

- Take $m^* = \frac{n}{3}$.
- Number of policies with $m^*$ or more improvable states visited

$$\leq \frac{2^n}{m^*} = \frac{2^n}{n/3}.$$ 

- Number of policies with fewer than $m^*$ improvable states visited
Howard’s Policy Iteration (2-action MDPs)

- Take $m^* = \frac{n}{3}$.
- Number of policies with $m^*$ or more improvable states visited
  
  \[
  \leq \frac{2^n}{m^*} = \frac{2^n}{n/3}.
  \]

- Number of policies with fewer than $m^*$ improvable states visited
  
  \[
  \leq \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{m^* - 1}
  \]
Howard’s Policy Iteration (2-action MDPs)

- Take $m^* = \frac{n}{3}$.
- Number of policies with $m^*$ or more improvable states visited
  \[
  \leq \frac{2^n}{m^*} = \frac{2^n}{n/3}.
  \]
- Number of policies with fewer than $m^*$ improvable states visited
  \[
  \leq \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{m^* - 1} \leq 3 \frac{2^n}{n}.
  \]
Howard’s Policy Iteration (2-action MDPs)

- Take $m^* = \frac{n}{3}$.
- Number of policies with $m^*$ or more improvable states visited

$$\leq \frac{2^n}{m^*} = \frac{2^n}{n/3}.$$ 

- Number of policies with fewer than $m^*$ improvable states visited

$$\leq \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{m^* - 1} \leq 3 \frac{2^n}{n}.$$ 

Number of iterations taken by Howard’s PI: $O\left(\frac{2^n}{n}\right)$ [MS99, HGDJ14].
Markov Decision Problems

1. Policy iteration: variants and complexity bounds

2. Analysis of bounds
   - Basic tools
   - Howard’s PI with $k = 2$
   - BSPI with $k = 2$
   - Open problems

3. Review of MDP planning
Batch-Switching Policy Iteration (BSPI) (2-action MDPs)

Howard’s Policy Iteration takes at most ____ iterations on a 2-state MDP!
Batch-Switching Policy Iteration (BSPI) (2-action MDPs)

Howard’s Policy Iteration takes at most 3 iterations on a 2-state MDP!
Batch-Switching Policy Iteration (BSPI) (2-action MDPs)

Howard’s Policy Iteration takes at most _3_ iterations on a 2-state MDP!
Batch-Switching Policy Iteration (BSPI)
Partition the states into 2-sized batches; arranged from left to right.
Given a policy, improve the rightmost set containing an improvable state.
Batch-Switching Policy Iteration (BSPI)

Partition the states into 2-sized batches; arranged from left to right. Given a policy, improve the rightmost set containing an improvable state.

$$\pi_1 \begin{array}{c|c|c|c|c|c|c|c} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 \end{array}$$

Left-most batch can change only when all other columns are non-improvable.

Left-most batch can change at most 3 times (following previous result).

$$T(n) \leq 3 \times T(n-2) \leq \sqrt{3} n.$$
Batch-Switching Policy Iteration (BSPI)

Partition the states into 2-sized batches; arranged from left to right.
Given a policy, improve the rightmost set containing an improvable state.
Batch-Switching Policy Iteration (BSPI)

Partition the states into 2-sized batches; arranged from left to right. Given a policy, improve the rightmost set containing an improvable state.

\(\pi_3\) 0 1 1 0 1 1 1 0 1 0

\(\pi_2\) 0 1 1 0 0 0 1 0 1 0

\(\pi_1\) 0 1 1 0 0 0 1 0 0 0

Left-most batch can change only when all other columns are non-improvable.

Left-most batch can change at most 3 times (following previous result).

\[ T(n) \leq 3 \times T(n-2) \leq \sqrt{3} n. \]
# Batch-Switching Policy Iteration (BSPI)

Partition the states into 2-sized batches; arranged from left to right. Given a policy, improve the rightmost set containing an improvable state.

<table>
<thead>
<tr>
<th>( \pi_4 )</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi_3 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \pi_2 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \pi_1 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Left-most batch can change only when all other columns are non-improvable.

Left-most batch can change at most 3 times (following previous result).

\[ T(n) \leq 3 \times T(n - 2) \leq \sqrt{3n} \]
**Batch-Switching Policy Iteration (BSPI)**

Partition the states into 2-sized batches; arranged from left to right. Given a policy, improve the **rightmost** set containing an **improvable** state.

<table>
<thead>
<tr>
<th>States</th>
<th>$\pi_1$</th>
<th>$\pi_2$</th>
<th>$\pi_3$</th>
<th>$\pi_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_4$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$s_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_6$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$s_7$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_8$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_9$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_{10}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Left-most batch can change only when all other columns are non-improvable.
Batch-Switching Policy Iteration (BSPI)

Partition the states into 2-sized batches; arranged from left to right. Given a policy, improve the rightmost set containing an improvable state.

<table>
<thead>
<tr>
<th>$\pi_4$</th>
<th>$\pi_3$</th>
<th>$\pi_2$</th>
<th>$\pi_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>$s_2$</td>
<td>$s_3$</td>
<td>$s_4$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Left-most batch can change only when all other columns are non-improvable.
- Left-most batch can change at most 3 times (following previous result).
## Batch-Switching Policy Iteration (BSPI)

Partition the states into 2-sized batches; arranged from left to right. Given a policy, improve the rightmost set containing an improvable state.

- \( \pi_4 \):
  - 0 1 1 0 1 1 1 1 1 0

- \( \pi_3 \):
  - 0 1 1 0 1 1 1 0 1 0

- \( \pi_2 \):
  - 0 1 1 0 0 0 1 0 1 0

- \( \pi_1 \):
  - 0 1 1 0 0 0 1 0 0 0

### Observations
- Left-most batch can change only when all other columns are non-improvable.
- Left-most batch can change at most 3 times (following previous result).
- \( T(n) \leq 3 \times T(n - 2) \leq \sqrt{3}^n \).
Batch-Switching Policy Iteration (BSPI)

Howard’s Policy Iteration takes at most 5 iterations on a 3-state MDP!
Batch-Switching Policy Iteration (BSPI)

Howard’s Policy Iteration takes at most 5 iterations on a 3-state MDP!
Batch-Switching Policy Iteration (BSPI)

Howard’s Policy Iteration takes at most 5 iterations on a 3-state MDP!

The structures above are called Trajectory-bounding Trees (TBTs) [KMG16a] (and correspond to the Order Regularity Problem [H12, GHDJ15]).
Batch-Switching Policy Iteration (BSPI)

Howard's Policy Iteration takes at most 5 iterations on a 3-state MDP!

The structures above are called Trajectory-bounding Trees (TBTs) [KMG16a] (and correspond to the Order Regularity Problem [H12, GHDJ15]).

BSPI with 3-sized batches gives $T(n) \leq 5 \times T(n - 3) \leq 1.71^n$. 
## Upper Bounds

<table>
<thead>
<tr>
<th>Batch size</th>
<th>Depth of TBT</th>
<th>Bound on number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$2^n$</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>$1.7321^n$</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>$1.7100^n$</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>$1.6818^n$</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>$1.6703^n$</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>$1.6611^n$</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>$1.6479^n$</td>
</tr>
</tbody>
</table>

Depth of TBT for batch size 7 due to Gerencsérl et al. \([\text{GHDJ15}]\).

Will the bound continue to be non-increasing in the batch size? If so, $1.6479^n$ would be an upper bound for Howard’s Policy Iteration!
### Upper Bounds

<table>
<thead>
<tr>
<th>Batch size</th>
<th>Depth of TBT</th>
<th>Bound on number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$2^n$</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>$1.7321^n$</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>$1.7100^n$</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>$1.6818^n$</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>$1.6703^n$</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>$1.6611^n$</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>$1.6479^n$</td>
</tr>
</tbody>
</table>

Depth of TBT for batch size 7 due to Gerencsér et al. [GHDJ15].
## UpperBounds

<table>
<thead>
<tr>
<th>Batch size</th>
<th>Depth of TBT</th>
<th>Bound on number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$2^n$</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>$1.7321^n$</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>$1.7100^n$</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>$1.6818^n$</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>$1.6703^n$</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>$1.6611^n$</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>$1.6479^n$</td>
</tr>
</tbody>
</table>

Depth of TBT for batch size 7 due to Gerencsér et al. [GHDJ15].

Will the bound continue to be non-increasing in the batch size?
## Upper Bounds

<table>
<thead>
<tr>
<th>Batch size</th>
<th>Depth of TBT</th>
<th>Bound on number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$2^n$</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>$1.7321^n$</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>$1.7100^n$</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>$1.6818^n$</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>$1.6703^n$</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>$1.6611^n$</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>$1.6479^n$</td>
</tr>
</tbody>
</table>

Depth of TBT for batch size 7 due to Gerencsér et al. [GHDJ15].

**Will the bound continue to be non-increasing in the batch size?**

If so, $1.6479^n$ would be an upper bound for Howard’s Policy Iteration!
Averaged over $n$-state, 2-action MDPs with randomly generated transition and reward functions. Each point is an average over 100 randomly-generated MDP instances and initial policies [KMG16a].
Markov Decision Problems

1. Policy iteration: variants and complexity bounds

2. Analysis of bounds
   - Basic tools
   - Howard’s PI with $k = 2$
   - BSPI with $k = 2$
   - Open problems

3. Review of MDP planning
Open Problems

- Is the complexity of Howard’s PI on 2-action MDPs upper-bounded by the Fibonacci sequence ($\approx 1.6181^n$)?
- Is Howard’s PI the most efficient among deterministic PI algorithms (worst case over all MDPs)?
- Is there a super-linear lower bound on the number of iterations taken by Howard’s PI on 2-action MDPs?
- Is Howard’s PI strongly polynomial on deterministic MDPs?
- Is there a variant of PI that can visit all $k^n$ policies in some $n$-state, $k$-action MDP—implying an $\Omega(k^n)$ lower bound?
- Is there a strongly polynomial algorithm for MDP planning?
Markov Decision Problems

1. Policy iteration: variants and complexity bounds

2. Analysis of bounds
   - Basic tools
   - Howard’s PI with $k = 2$
   - BSPI with $k = 2$
   - Open problems

3. Review of MDP planning
Summary of MDP Planning

- MDPs are an abstraction of sequential decision making.
- Many applications; many different formulations.
- Essential solution concept: optimal policy (known to exist).

- Three main families of planning algorithms: value iteration, linear programming, policy iteration.
- Have strengths and weaknesses in theory and in practice. Can combine.

- We showed correctness of all three methods.
- Used Banach’s fixed-point theorem, Bellman (optimality) operator.

- What if $T, R$ were not given, but have to be learned from interaction? Can we still learn to act optimally?
- Yes: that’s the reinforcement learning problem. Next class!