Theoretical Analysis of Policy Iteration

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
shivaram@cse.iitb.ac.in

August 2017
Overview

1. Background
   MDP Planning
   Bellman’s Equations and Bellman’s Optimality Equations
   Solution strategies
   Strong Running-time Bounds

2. Policy Iteration
   Policy Improvement
   Proof of Policy Improvement Theorem
   Policy Iteration algorithm
   Switching strategies and bounds

3. Analysis of Policy Iteration on 2-action MDPs
   Basic Tools and Results
   Howard’s Policy Iteration
   Mansour and Singh’s Randomised Policy Iteration
   Batch-Switching Policy Iteration

4. Summary and Outlook
   Results for $k$-action MDPs
   Open problems
   References
   Conclusion
Overview

1. **Background**
   - MDP Planning
   - Bellman’s Equations and Bellman’s Optimality Equations
   - Solution strategies
   - Strong Running-time Bounds

2. **Policy Iteration**
   - Policy Improvement
   - Proof of Policy Improvement Theorem
   - Policy Iteration algorithm
   - Switching strategies and bounds

3. **Analysis of Policy Iteration on 2-action MDPs**
   - Basic Tools and Results
   - Howard’s Policy Iteration
   - Mansour and Singh’s Randomised Policy Iteration
   - Batch-Switching Policy Iteration

4. **Summary and Outlook**
   - Results for $k$-action MDPs
   - Open problems
   - References
   - Conclusion
MDP Planning

- Markov Decision Problem: general abstraction of sequential decision making.

- An MDP comprises a tuple \((S, A, R, T, \gamma)\), where
  
  - \(S\) is a set of states (with \(|S| = n\)),
  
  - \(A\) is a set of actions (with \(|A| = k\)),
  
  - \(R(s, a)\) is a bounded real number, \(\forall s \in S, \forall a \in A\), and
  
  - \(T(s, a)\) is a probability distribution over \(S\), \(\forall s \in S, \forall a \in A\).

- A policy \(\pi : S \rightarrow A\) specifies an action from each state, and yields trajectory
  
  \[ s^0, a^0 = \pi(s^0), r^0, s^1, a^1 = \pi(s^1), r^1, s^2, \ldots \]

- The value of a policy \(\pi\) from state \(s\) is:

  \[
  V^\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r^t \mid s^0 = s, a^t = \pi(s^t), t = 0, 1, 2, \ldots \right],
  \]

  where \(\gamma \in [0, 1)\) is a discount factor.
MDP Planning

- Markov Decision Problem: general abstraction of sequential decision making.

- An MDP comprises a tuple \((S, A, R, T, \gamma)\), where
  - \(S\) is a set of states (with \(|S| = n\)),
  - \(A\) is a set of actions (with \(|A| = k\)),
  - \(R(s, a)\) is a bounded real number, \(\forall s \in S, \forall a \in A\), and
  - \(T(s, a)\) is a probability distribution over \(S\), \(\forall s \in S, \forall a \in A\).

- A policy \(\pi : S \rightarrow A\) specifies an action from each state, and yields trajectory
  \[s^0, a^0 = \pi(s^0), r^0, s^1, a^1 = \pi(s^1), r^1, s^2, \ldots\]

- The value of a policy \(\pi\) from state \(s\) is:
  \[V^\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r^t \mid s^0 = s, a^t = \pi(s^t), t = 0, 1, 2, \ldots \right]\]
  where
  \[\gamma \in [0, 1)\] is a discount factor.

**Planning problem**: Given \(S, A, R, T, \gamma\), find a policy \(\pi^*\) from the set of all policies \(\Pi\) such that \(\forall s \in S, \forall \pi \in \Pi: V^{\pi^*}(s) \geq V^\pi(s)\).
MDP Planning

- Markov Decision Problem: general abstraction of sequential decision making.

- An MDP comprises a tuple \((S, A, R, T, \gamma)\), where
  - \(S\) is a set of states (with \(|S| = n\)),
  - \(A\) is a set of actions (with \(|A| = k\)),
  - \(R(s, a)\) is a bounded real number, \(\forall s \in S, \forall a \in A\), and
  - \(T(s, a)\) is a probability distribution over \(S\), \(\forall s \in S, \forall a \in A\).

- A policy \(\pi : S \rightarrow A\) specifies an action from each state, and yields trajectory
  \[s^0, a^0 = \pi(s^0), r^0, s^1, a^1 = \pi(s^1), r^1, s^2, \ldots\]

- The value of a policy \(\pi\) from state \(s\) is:
  \[V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^t \mid s^0 = s, a^t = \pi(s^t), t = 0, 1, 2, \ldots\right]\]
  where \(\gamma \in [0, 1)\) is a discount factor.

**Planning problem:** Given \(S, A, R, T, \gamma\), find a policy \(\pi^*\) from the set of all policies \(\Pi\) such that \(\forall s \in S, \forall \pi \in \Pi: V^\pi^*(s) \geq V^\pi(s)\).
MDP Planning

- **Markov Decision Problem**: general abstraction of sequential decision making.

- An MDP comprises a tuple \((S, A, R, T, \gamma)\), where
  - \(S\) is a set of states (with \(|S| = n\)),
  - \(A\) is a set of actions (with \(|A| = k\), \(\leftarrow\) (We’ll specially consider \(k = 2\).)
  - \(R(s, a)\) is a bounded real number, \(\forall s \in S, \forall a \in A\), and
  - \(T(s, a)\) is a probability distribution over \(S\), \(\forall s \in S, \forall a \in A\).

- A policy \(\pi: S \rightarrow A\) specifies an action from each state, and yields trajectory
  \[s^0, a^0 = \pi(s^0), r^0, s^1, a^1 = \pi(s^1), r^1, s^2, \ldots.\]

- The value of a policy \(\pi\) from state \(s\) is:
  \[V^\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t r^t | s^0 = s, a^t = \pi(s^t), t = 0, 1, 2, \ldots \right],\]
  where \(\gamma \in [0, 1)\) is a discount factor.

**Planning problem**: Given \(S, A, R, T, \gamma\), find a policy \(\pi^*\) from the set of all policies \(\Pi\) such that \(\forall s \in S, \forall \pi \in \Pi: V^{\pi^*}(s) \geq V^{\pi}(s)\).
Illustration: MDPs as State Transition Diagrams

Notation: "transition probability, reward" marked on each arrow

States: $s_1$, $s_2$, $s_3$, and $s_4$.

Actions: Red (solid lines) and blue (dotted lines).

Transitions: Red action leads to same state with 20% chance, to next-clockwise state with 80% chance. Blue action leads to next-clockwise state or 2-removed-clockwise state with equal (50%) probability.

Rewards: $R(\ast, \ast, s_1) = 0$, $R(\ast, \ast, s_2) = 1$, $R(\ast, \ast, s_3) = -1$, $R(\ast, \ast, s_4) = 2$.

Discount factor: $\gamma = 0.9$. 
Bellman’s Equations

Recall: \( V^\pi(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots | s^0 = s, a^t = \pi(s^t) \text{ for } t = 0, 1, \ldots ] \).

Bellman’s Equations: \( \forall s \in S \),

\[
V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s').
\]

\( V^\pi : S \to \mathbb{R} \) is called the value function of \( \pi \).
Bellman’s Equations

Recall: $V^\pi(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots | s^0 = s, a^t = \pi(s^t) \text{ for } t = 0, 1, \ldots]$. 

**Bellman’s Equations:** $\forall s \in S,$ 
\[
V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s').
\]

$V^\pi : S \to \mathbb{R}$ is called the value function of $\pi$.

Define: $\forall s \in S, \forall a \in A,$ 
\[
Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^\pi(s').
\]

$Q^\pi$ is called the action value function of $\pi$.

Observe that $V^\pi(s) = Q^\pi(s, \pi(s))$. 
Bellman’s Equations

Recall: \( V^\pi(s) = \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots | s_0 = s, a^t = \pi(s^t) \text{ for } t = 0, 1, \ldots] \).

Bellman’s Equations: \( \forall s \in S \),

\[
V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s').
\]

\( V^\pi : S \rightarrow \mathbb{R} \) is called the value function of \( \pi \).

Define: \( \forall s \in S, \forall a \in A \),

\[
Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^\pi(s').
\]

\( Q^\pi \) is called the action value function of \( \pi \).

Observe that \( V^\pi(s) = Q^\pi(s, \pi(s)) \).

The variables in Bellman’s Equations are the elements of \( V^\pi \).

\( n \) linear equations in \( n \) unknowns.
Bellman’s Equations

- Recall: $V^\pi(s) = \mathbb{E}[r^0 + \gamma r^1 + \gamma^2 r^2 + \ldots | s^0 = s, a^t = \pi(s^t) \text{ for } t = 0, 1, \ldots]$. 

  **Bellman’s Equations:** $\forall s \in S,$

  

  $V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s').$

  

  $V^\pi : S \rightarrow \mathbb{R}$ is called the **value function** of $\pi$.

- Define: $\forall s \in S, \forall a \in A,$

  

  $Q^\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^\pi(s').$

  

  $Q^\pi$ is called the **action value function** of $\pi$.

  Observe that $V^\pi(s) = Q^\pi(s, \pi(s))$.

- The variables in Bellman’s Equations are the elements of $V^\pi$. 

  $n$ linear equations in $n$ unknowns.

---

Given $S, A, T, R, \gamma,$ and a fixed policy $\pi$, we can solve Bellman’s Equations to obtain $V^\pi$ and $Q^\pi$. This step is called **Policy Evaluation**.
Bellman’s Optimality Equations

- The **Optimal Value Function** \( V^* \) is unique solution of: \( \forall s \in S, \)

\[
V^*(s) = \max_{a \in A} \left( R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right).
\]

These are Bellman’s Optimality Equations.

- The **Optimal Action Value Function** \( Q^* \) is given by: \( \forall s \in S, \forall a \in A, \)

\[
Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s').
\]

- Given \( Q^* \), we may obtain \( \pi^* \) by setting, \( \forall s \in S: \)

\[
\pi^*(s) \leftarrow \arg\max_{a \in A} Q^*(s, a).
\]

Given \( \pi^* \), how can we obtain \( V^* \) and \( Q^* \)?
Bellman’s Optimality Equations

- The **Optimal Value Function** \( V^* \) is unique solution of: \( \forall s \in S, \)

\[
V^*(s) = \max_{a \in A} \left( R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right).
\]

These are **Bellman’s Optimality Equations**.

- The **Optimal Action Value Function** \( Q^* \) is given by: \( \forall s \in S, \forall a \in A, \)

\[
Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s').
\]

- Given \( Q^* \), we may obtain \( \pi^* \) by setting, \( \forall s \in S: \)

\[
\pi^*(s) \leftarrow \arg\max_{a \in A} Q^*(s, a).
\]

Given \( \pi^* \), how can we obtain \( V^* \) and \( Q^* \)? By **policy evaluation** (previous slide).
Solution Strategies

- **Value Iteration**

\[ V_0 \leftarrow \text{Arbitrary, element-wise bounded, } n\text{-length vector. } t \leftarrow 0. \]

**Repeat:**

For \( s \in S \):

\[ V_{t+1}(s) \leftarrow \max_{a \in A} \left( R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V_t(s') \right). \]

\[ t \leftarrow t + 1. \]

**Until** \( V_t \approx V_{t-1} \) (up to machine precision).

Convergence to \( V^* \) guaranteed using a max-norm contraction argument.

- **Linear Programming**

\[
\text{Minimise } \sum_{s \in S} V(s) \\
\text{subject to } V(s) \geq \left( R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s') \right), \forall s \in S, \forall a \in A.
\]

\( n \text{ variables, } nk \text{ constraints (or dual with } nk \text{ variables, } n \text{ constraints).} \)
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \[\text{poly}(n, k, B, \frac{1}{1-\gamma})\], where \(B\) is the number of bits used to represent the MDP.
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \( \text{poly}(n, k, B, \frac{1}{1-\gamma}) \), where \( B \) is the number of bits used to represent the MDP. Not a strong bound.
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \[ \text{poly}(n, k, B, \frac{1}{1-\gamma}) \], where \( B \) is the number of bits used to represent the MDP. Not a strong bound.

- Strong bounds depend solely on \( n \) and \( k \) (no dependence on \( B, \gamma \), etc.).
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]: \( \text{poly}(n, k, B, \frac{1}{1-\gamma}) \), where \( B \) is the number of bits used to represent the MDP. Not a strong bound.

- Strong bounds depend solely on \( n \) and \( k \) (no dependence on \( B, \gamma, \) etc.).
  
  Is there a strong upper bound on the complexity of policy evaluation?
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \[ \text{poly}(n, k, B, \frac{1}{1-\gamma}) \], where \( B \) is the number of bits used to represent the MDP. Not a strong bound.

- Strong bounds depend solely on \( n \) and \( k \) (no dependence on \( B, \gamma, \) etc.).
  Is there a strong upper bound on the complexity of policy evaluation? \( O(n^2 k + n^3) \).
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \[ \text{poly}(n, k, B, \frac{1}{1-\gamma}) \], where \( B \) is the number of bits used to represent the MDP.
  Not a strong bound.

- Strong bounds depend solely on \( n \) and \( k \) (no dependence on \( B, \gamma \), etc.).
  Is there a strong upper bound on the complexity of policy evaluation? \( O(n^2 k + n^3) \).
  Can you give a strong bound on the running time of MDP planning?
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \[ \text{poly}(n, k, B \cdot \frac{1}{1-\gamma}) \], where \( B \) is the number of bits used to represent the MDP.
  Not a strong bound.

- Strong bounds depend solely on \( n \) and \( k \) (no dependence on \( B, \gamma \), etc.).

  Is there a strong upper bound on the complexity of \textit{policy evaluation}? \( O(n^2 k + n^3) \).
  Can you give a strong bound on the running time of MDP planning? \( \text{poly}(n, k) \cdot k^n \).
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \[ \text{poly}(n, k, B, \frac{1}{1-\gamma}) \], where \( B \) is the number of bits used to represent the MDP.
  Not a strong bound.

- Strong bounds depend solely on \( n \) and \( k \) (no dependence on \( B, \gamma, \) etc.).
  Is there a strong upper bound on the complexity of policy evaluation? \( O(n^2 k + n^3) \).
  Can you give a strong bound on the running time of MDP planning? \( \text{poly}(n, k) \cdot k^n \).

- Bounds for Linear Programming-type approaches to MDP planning:
  \( \text{poly}(n, k, B) \) [K80, K84].
  \( \text{poly}(n, k) \cdot \exp(O(\sqrt{n \log(n)})) \) (Expected) [MSW96].
  \( \text{poly}(n, k) \cdot k^{0.6834n} \) [GK17].
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \[ \text{poly}(n, k, B, \frac{1}{1-\gamma}) \], where \( B \) is the number of bits used to represent the MDP.
  Not a strong bound.

- Strong bounds depend solely on \( n \) and \( k \) (no dependence on \( B, \gamma \), etc.).
  Is there a strong upper bound on the complexity of policy evaluation? \( O(n^2 k + n^3) \).
  Can you give a strong bound on the running time of MDP planning? \( \text{poly}(n, k) \cdot k^n \).

- Bounds for Linear Programming-type approaches to MDP planning:
  \( \text{poly}(n, k, B) \) [K80, K84].
  \( \text{poly}(n, k) \cdot \exp(O(\sqrt{n \log(n)})) \) (Expected) [MSW96].
  \( \text{poly}(n, k) \cdot k^{0.6834n} \) [GK17].
  \( \text{poly}(n, k) \) for deterministic MDPs [MTZ10, PY13].
Strong Running-time Bounds

- Computation model: Infinite precision arithmetic (or Real RAM) model.

- Upper Bound for Value Iteration [LDK95]:
  \(\text{poly}(n, k, B, \frac{1}{1-\gamma})\), where \(B\) is the number of bits used to represent the MDP. Not a strong bound.

- Strong bounds depend solely on \(n\) and \(k\) (no dependence on \(B, \gamma\), etc.).
  
  Is there a strong upper bound on the complexity of policy evaluation? \(O(n^2 k + n^3)\).
  
  Can you give a strong bound on the running time of MDP planning? \(\text{poly}(n, k) \cdot k^n\).

- Bounds for Linear Programming-type approaches to MDP planning:
  \(\text{poly}(n, k, B)\) [K80, K84].
  \(\text{poly}(n, k) \cdot \exp(\mathcal{O}(\sqrt{n \log(n)}))\) (Expected) [MSW96].
  \(\text{poly}(n, k) \cdot k^{0.6834n}\) [GK17].
  \(\text{poly}(n, k)\) for deterministic MDPs [MTZ10, PY13].

- Appeal of Policy Iteration:
  
  Theoretical: naturally yields strong bounds (also enjoys good weak bounds [P94]).
  
  Practical: very fast on MDPs encountered in typical applications.
Overview

1. Background
   MDP Planning
   Bellman’s Equations and Bellman’s Optimality Equations
   Solution strategies
   Strong Running-time Bounds

2. Policy Iteration
   Policy Improvement
   Proof of Policy Improvement Theorem
   Policy Iteration algorithm
   Switching strategies and bounds

3. Analysis of Policy Iteration on 2-action MDPs
   Basic Tools and Results
   Howard’s Policy Iteration
   Mansour and Singh’s Randomised Policy Iteration
   Batch-Switching Policy Iteration

4. Summary and Outlook
   Results for $k$-action MDPs
   Open problems
   References
   Conclusion
Policy Improvement

\[ \pi \]

\( S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7 \quad S_8 \)
Policy Improvement

\[ \pi \]

\( s_1 \quad s_2 \quad s_3 \quad s_4 \quad s_5 \quad s_6 \quad s_7 \quad s_8 \)
Policy Improvement

\[ Q^\pi(s_3, \blacktriangleleft) \leq Q^\pi(s_3, \blacktriangleright) \]
Policy Improvement

\[ Q^\pi(s, \ \square) \geq Q^\pi(s', \ \square) \]

\[ Q^\pi(s_3, \ \square) \leq Q^\pi(s_3', \ \square) \]
Policy Improvement

Improvable states
Policy Improvement

Improvable states

Improving actions

Shivaram Kalyanakrishnan (2017)
Theoretical Analysis of Policy Iteration
Given $\pi$, pick one or more improvable states, and in them, switch to an arbitrary improving action. Let the resulting policy be $\pi'$. 

![Improvable states and improving actions diagram]
Given $\pi$, pick one or more improvable states, and in them, switch to an arbitrary improving action. Let the resulting policy be $\pi'$. 
Policy Improvement

Given $\pi$, 
Pick one or more improvable states, and in them, 
Switch to an arbitrary improving action. 
Let the resulting policy be $\pi'$. 

**Policy Improvement Theorem:**
(1) If $\pi$ has no improvable states, then it is optimal, else 
(2) if $\pi'$ is obtained as above, then 
\[ \forall s \in S : V^{\pi'}(s) \geq V^\pi(s) \text{ and } \exists s \in S : V^{\pi'}(s) > V^\pi(s). \]
Policy Improvement

Given $\pi$,

Pick one or more improvable states, and in them,
Switch to an arbitrary improving action.
Let the resulting policy be $\pi'$.

Policy Improvement Theorem:
(1) If $\pi$ has no improvable states, then it is optimal, else
(2) if $\pi'$ is obtained as above, then
$$\forall s \in S : V^\pi'(s) \geq V^\pi(s) \text{ and } \exists s \in S : V^\pi'(s) > V^\pi(s).$$
Definitions and Basic Facts

For $X: S \rightarrow \mathbb{R}$ and $Y: S \rightarrow \mathbb{R}$, we define $X \succeq Y$ if $\forall s \in S : X(s) \geq Y(s)$, and we define $X \succ Y$ if $X \succeq Y$ and $\exists s \in S : X(s) > Y(s)$. 
Definitions and Basic Facts

For $X : S \rightarrow \mathbb{R}$ and $Y : S \rightarrow \mathbb{R}$, we define $X \succeq Y$ if $\forall s \in S : X(s) \geq Y(s)$, and we define $X \succ Y$ if $X \succeq Y$ and $\exists s \in S : X(s) > Y(s)$.

For policies $\pi_1, \pi_2 \in \Pi$, we define $\pi_1 \succeq \pi_2$ if $V^{\pi_1} \succeq V^{\pi_2}$, and we define $\pi_1 \succ \pi_2$ if $V^{\pi_1} \succ V^{\pi_2}$. 
Definitions and Basic Facts

For $X : S \to \mathbb{R}$ and $Y : S \to \mathbb{R}$, we define $X \succeq Y$ if $\forall s \in S : X(s) \geq Y(s)$, and we define $X \succ Y$ if $X \succeq Y$ and $\exists s \in S : X(s) > Y(s)$.

For policies $\pi_1, \pi_2 \in \Pi$, we define $\pi_1 \succeq \pi_2$ if $V^{\pi_1} \succeq V^{\pi_2}$, and we define $\pi_1 \succ \pi_2$ if $V^{\pi_1} \succ V^{\pi_2}$.

**Bellman Operator.** For $\pi \in \Pi$, we define $B^\pi : (S \to \mathbb{R}) \to (S \to \mathbb{R})$ as follows: for $X : S \to \mathbb{R}$ and $\forall s \in S$,

$$(B^\pi (X))(s) \overset{\text{def}}{=} R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') X(s').$$
Definitions and Basic Facts

- For $X: S \rightarrow \mathbb{R}$ and $Y: S \rightarrow \mathbb{R}$, we define $X \succeq Y$ if $\forall s \in S: X(s) \geq Y(s)$, and we define $X \succ Y$ if $X \succeq Y$ and $\exists s \in S: X(s) > Y(s)$.

- For policies $\pi_1, \pi_2 \in \Pi$, we define $\pi_1 \succeq \pi_2$ if $V^{\pi_1} \succeq V^{\pi_2}$, and we define $\pi_1 \succ \pi_2$ if $V^{\pi_1} \succ V^{\pi_2}$.

- **Bellman Operator.** For $\pi \in \Pi$, we define $B^\pi : (S \rightarrow \mathbb{R}) \rightarrow (S \rightarrow \mathbb{R})$ as follows: for $X: S \rightarrow \mathbb{R}$ and $\forall s \in S$,

  $$(B^\pi(X))(s) \overset{\text{def}}{=} R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') X(s').$$

- **Fact 1.** For $\pi \in \Pi$, $X: S \rightarrow \mathbb{R}$, and $Y: S \rightarrow \mathbb{R}$:

  if $X \succeq Y$, then $B^\pi(X) \succeq B^\pi(Y)$. 

Shivaram Kalyanakrishnan (2017) Theoretical Analysis of Policy Iteration
Definitions and Basic Facts

- For $X : S \to \mathbb{R}$ and $Y : S \to \mathbb{R}$, we define $X \succeq Y$ if $\forall s \in S : X(s) \geq Y(s)$, and we define $X \succ Y$ if $X \succeq Y$ and $\exists s \in S : X(s) > Y(s)$.

For policies $\pi_1, \pi_2 \in \Pi$, we define $\pi_1 \succeq \pi_2$ if $V^{\pi_1} \succeq V^{\pi_2}$, and we define $\pi_1 \succ \pi_2$ if $V^{\pi_1} \succ V^{\pi_2}$.

- **Bellman Operator.** For $\pi \in \Pi$, we define $B^\pi : (S \to \mathbb{R}) \to (S \to \mathbb{R})$ as follows: for $X : S \to \mathbb{R}$ and $\forall s \in S$,

  $$(B^\pi (X))(s) \overset{\text{def}}{=} R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') X(s').$$

- **Fact 1.** For $\pi \in \Pi$, $X : S \to \mathbb{R}$, and $Y : S \to \mathbb{R}$:

  if $X \succeq Y$, then $B^\pi (X) \succeq B^\pi (Y)$.

- **Fact 2.** For $\pi \in \Pi$ and $X : S \to \mathbb{R}$:

  $$\lim_{l \to \infty} (B^\pi)^l (X) = V^\pi.$$
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi$, $\forall s \in S$: $B^{\pi'}(V^{\pi})(s) = Q^{\pi}(s, \pi'(s))$. 
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi, \forall s \in S$: $B^{\pi'}(V^{\pi})(s) = Q^{\pi}(s, \pi'(s))$.

$\pi$ has no improvable states

$\implies \forall \pi' \in \Pi : V^{\pi} \succeq B^{\pi'}(V^{\pi})$
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi$, $\forall s \in S$: $B^{\pi'}(V^\pi)(s) = Q^\pi(s, \pi'(s))$.

$\pi$ has no improvable states

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi)$
Proof of Policy Improvement Theorem

Observe that for \( \pi, \pi' \in \Pi, \forall s \in S: B^{\pi'}(V^\pi)(s) = Q^\pi(s, \pi'(s)). \)

\( \pi \) has no improvable states

\[
\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi)
\]

\[
\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi)
\]

\[
\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi) \succeq \cdots \succeq \lim_{l \to \infty} (B^{\pi'})^l(V^\pi)
\]
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi, \forall s \in S$: $B^{\pi'}(V^\pi)(s) = Q^\pi(s, \pi'(s))$.

$\pi$ has no improvable states

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi) \succeq \cdots \succeq \lim_{l \to \infty} (B^{\pi'})^l(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq V^{\pi'}$. 
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi, \forall s \in S: B^\pi'(V^\pi)(s) = Q^\pi(s, \pi'(s))$.

$\pi$ has no improvable states

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^\pi'(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^\pi'(V^\pi) \succeq (B^\pi')^2(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^\pi'(V^\pi) \succeq (B^\pi')^2(V^\pi) \succeq \cdots \succeq \lim_{l \to \infty} (B^\pi')^l(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq V^{\pi'}$.

$\pi$ has improvable states and policy improvement yields $\pi'$
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi, \forall s \in S: B_{\pi'}^{\pi'}(V^\pi)(s) = Q^\pi(s, \pi'(s))$.

$\pi$ has no improvable states

$\implies \forall \pi' \in \Pi : V^\pi \succeq B_{\pi'}^{\pi'}(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B_{\pi'}^{\pi'}(V^\pi) \succeq (B_{\pi'}^{\pi'})^2(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B_{\pi'}^{\pi'}(V^\pi) \succeq (B_{\pi'}^{\pi'})^2(V^\pi) \succeq \cdots \succeq \lim_{l \to \infty} (B_{\pi'}^{\pi'})^l(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq V^\pi'$.

$\pi$ has improvable states and policy improvement yields $\pi'$

$\implies B_{\pi'}^{\pi'}(V^\pi) \succ V^\pi$
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi$, $\forall s \in S$: $B^{\pi'}(V^\pi)(s) = Q^\pi(s, \pi'(s))$.

$\pi$ has no improvable states

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi) \succeq \cdots \succeq \lim_{l \to \infty} (B^{\pi'})^l(V^\pi)$

$\implies \forall \pi' \in \Pi : V^\pi \succeq V^{\pi'}$.

$\pi$ has improvable states and policy improvement yields $\pi'$

$\implies B^{\pi'}(V^\pi) \succ V^\pi$

$\implies (B^{\pi'})^2(V^\pi) \succeq B^{\pi'}(V^\pi) \succ V^\pi$
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi, \forall s \in S: B^\pi'(V^\pi)(s) = Q^\pi(s, \pi'(s))$.

$\pi$ has no improvable states

$\implies \forall \pi' \in \Pi: V^\pi \succeq B^\pi'(V^\pi) \implies B^\pi'(V^\pi) \succeq (B^\pi')^2(V^\pi) \implies \cdots \succeq \lim_{l \to \infty} (B^\pi')^l(V^\pi) \implies \forall \pi' \in \Pi: V^\pi \succeq V^\pi'$.

$\pi$ has improvable states and policy improvement yields $\pi'$

$\implies B^\pi'(V^\pi) \succ V^\pi \implies (B^\pi')^2(V^\pi) \succeq B^\pi'(V^\pi) \succ V^\pi \implies \cdots \succeq (B^\pi')^2(V^\pi) \succeq B^\pi'(V^\pi) \succ V^\pi \implies \lim_{l \to \infty} (B^\pi')^l(V^\pi) \succeq \cdots \succeq (B^\pi')^2(V^\pi) \succeq B^\pi'(V^\pi) \succ V^\pi$. 
Proof of Policy Improvement Theorem

Observe that for $\pi, \pi' \in \Pi$, $\forall s \in S$: $B^{\pi'}(V^\pi)(s) = Q^\pi(s, \pi'(s))$.

$\pi$ has no improvable states

$$\Rightarrow \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi)$$

$$\Rightarrow \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi)$$

$$\Rightarrow \forall \pi' \in \Pi : V^\pi \succeq B^{\pi'}(V^\pi) \succeq (B^{\pi'})^2(V^\pi) \succeq \cdots \succeq \lim_{l \to \infty} (B^{\pi'})^l(V^\pi)$$

$$\Rightarrow \forall \pi' \in \Pi : V^\pi \succeq V^{\pi'}.$$

$\pi$ has improvable states and policy improvement yields $\pi'$

$$\Rightarrow B^{\pi'}(V^\pi) \succ V^\pi$$

$$\Rightarrow (B^{\pi'})^2(V^\pi) \succeq B^{\pi'}(V^\pi) \succ V^\pi$$

$$\Rightarrow \lim_{l \to \infty} (B^{\pi'})^l(V^\pi) \succeq \cdots \succeq (B^{\pi'})^2(V^\pi) \succeq B^{\pi'}(V^\pi) \succ V^\pi$$

$$\Rightarrow V^{\pi'} \succ V^\pi.$$
Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.

While $\pi$ has improvable states:

$\pi \leftarrow$ PolicyImprovement($\pi$).
Policy Iteration Algorithm

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

\textbf{While} \ \pi \ \text{has improvable states:}

\[
\pi \leftarrow \text{PolicyImprovement}(\pi).
\]
Policy Iteration Algorithm

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

\[
\textbf{While } \pi \text{ has improvable states:}
\]

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

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

\( \pi \leftarrow \text{PolicyImprovement}(\pi) \).
Policy Iteration Algorithm

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

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

\( \pi \leftarrow \text{PolicyImprovement}(\pi) \).
Policy Iteration Algorithm

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

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

\[ \pi \leftarrow \text{PolicyImprovement}(\pi). \]
Policy Iteration Algorithm

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

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

\[ \pi \leftarrow \text{PolicyImprovement}(\pi). \]
## 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 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 Randomised PI [MS99]</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
<td>$\approx O\left(\left(\frac{k}{2}\right)^n\right)$</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>$k = 2$</th>
<th>General $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard’s 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 Randomised PI [MS99]</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
<td>$\approx O\left(\left(\frac{k}{2}\right)^n\right)$</td>
</tr>
<tr>
<td>Batch-switching PI (BSPI) [KMG16a]</td>
<td>Deterministic</td>
<td>$1.6479^n$</td>
<td>–</td>
</tr>
<tr>
<td>Recursive BSPI [GK17]</td>
<td>Deterministic</td>
<td>–</td>
<td>$k^{0.7207n}$</td>
</tr>
<tr>
<td>Recursive Simple PI [KMG16b]</td>
<td>Randomised</td>
<td>–</td>
<td>$(2 + \ln(k - 1))^n$</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>$k = 2$</th>
<th>General $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard’s PI</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>[H60, MS99]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mansour and Singh’s Randomised PI</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
<td>$\approx O\left((\frac{k}{2})^n\right)$</td>
</tr>
<tr>
<td>[MS99]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch-switching PI (BSPI) [KMG16a]</td>
<td>Deterministic</td>
<td>$1.6479^n$</td>
<td>–</td>
</tr>
<tr>
<td>Recursive BSPI [GK17]</td>
<td>Deterministic</td>
<td>–</td>
<td>$k^{0.7207n}$</td>
</tr>
<tr>
<td>Recursive Simple PI [KMG16b]</td>
<td>Randomised</td>
<td>–</td>
<td>$(2 + \ln(k - 1))^n$</td>
</tr>
</tbody>
</table>

#### Lower bounds on number of iterations

$\Omega\left(2^{n/7}\right)$ Howard’s PI on $n$-state MDPs with $\Theta(n)$ actions per state [F10, HGD12].
### Upper bounds on number of iterations

<table>
<thead>
<tr>
<th>PI Variant</th>
<th>Type</th>
<th>Type $k = 2$</th>
<th>General $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Howard’s 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 Randomised PI [MS99]</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
<td>$\approx O \left( \left( \frac{k}{2} \right)^n \right)$</td>
</tr>
<tr>
<td>Batch-switching PI (BSPI) [KMG16a]</td>
<td>Deterministic</td>
<td>$1.6479^n$</td>
<td>–</td>
</tr>
<tr>
<td>Recursive BSPI [GK17]</td>
<td>Deterministic</td>
<td>–</td>
<td>$k^{0.7207n}$</td>
</tr>
<tr>
<td>Recursive Simple PI [KMG16b]</td>
<td>Randomised</td>
<td>–</td>
<td>$(2 + \ln(k - 1))^n$</td>
</tr>
</tbody>
</table>

### Lower bounds on number of iterations

- $\Omega(2^{n/7})$ Howard’s PI on $n$-state MDPs with $\Theta(n)$ actions per state [F10, HGD12].
- $\Omega(2^{n/2})$ Simple PI on $n$-state, 2-action MDPs [MC94].
### 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 PI</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>[H60, MS99]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mansour and Singh’s Randomised PI [MS99]</td>
<td>Randomised</td>
<td>$1.7172^n$</td>
<td>$\approx O\left(\left(\frac{k}{2}\right)^n\right)$</td>
</tr>
<tr>
<td>Batch-switching PI (BSPI) [KMG16a]</td>
<td>Deterministic</td>
<td>$1.6479^n$</td>
<td></td>
</tr>
<tr>
<td>Recursive BSPI [GK17]</td>
<td>Deterministic</td>
<td>–</td>
<td>$k^{0.7207n}$</td>
</tr>
<tr>
<td>Recursive Simple PI [KMG16b]</td>
<td>Randomised</td>
<td>–</td>
<td>$(2 + \ln(k - 1))^n$</td>
</tr>
</tbody>
</table>

#### Lower bounds on number of iterations

- $\Omega\left(\frac{2^n}{n}\right)$: Howard’s PI on $n$-state MDPs with $\Theta(n)$ actions per state [F10, HGD12].
- $\Omega\left(\frac{2^n}{2}\right)$: Simple PI on $n$-state, 2-action MDPs [MC94].
- $\Omega(n)$: Howard’s PI on $n$-state, 2-action MDPs [HZ10].
Overview

1. **Background**
   - MDP Planning
   - Bellman’s Equations and Bellman’s Optimality Equations
   - Solution strategies
   - Strong Running-time Bounds

2. **Policy Iteration**
   - Policy Improvement
   - Proof of Policy Improvement Theorem
   - Policy Iteration algorithm
   - Switching strategies and bounds

3. **Analysis of Policy Iteration on 2-action MDPs**
   - Basic Tools and Results
   - Howard’s Policy Iteration
   - Mansour and Singh’s Randomised Policy Iteration
   - Batch-Switching Policy Iteration

4. **Summary and Outlook**
   - Results for $k$-action MDPs
   - Open problems
   - References
   - Conclusion
Basic Tool: Policy Improvement and Policy Deproofment

\[ \pi' \succ \pi. \]

Policy Improvement

Shivaram Kalyanakrishnan (2017)
Basic Tool: Policy Improvement and Policy Deprovement

\[ \pi' \succ \pi. \]

Shivaram Kalyanakrishnan (2017)

Theoretical Analysis of Policy Iteration
Consider $\pi, \pi' \in \Pi$. If $V^\pi \neq V^{\pi'}$, then $\pi$ and $\pi'$ cannot have the same set of improvable states.
Consider $\pi, \pi' \in \Pi$. If $V^\pi \neq V^{\pi'}$, then $\pi$ and $\pi'$ cannot have the same set of improvable states.
Consider \( \pi, \pi' \in \Pi \). If \( V^\pi \neq V^{\pi'} \), then \( \pi \) and \( \pi' \) cannot have the same set of improvable states.
Basic Tool: Property of Improvement sets in 2-action MDPs

Consider \( \pi, \pi' \in \Pi \). If \( V^\pi \neq V^{\pi'} \), then \( \pi \) and \( \pi' \) cannot have the same set of improvable states.

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

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

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

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]
Consider $\pi, \pi' \in \Pi$. If $V^\pi \neq V^{\pi'}$, then $\pi$ and $\pi'$ cannot have the same set of improvable states.
Consider $\pi, \pi' \in \Pi$. If $V^\pi \neq V^{\pi'}$, then $\pi$ and $\pi'$ cannot have the same set of improvable states.
Consider $\pi, \pi' \in \Pi$. If $V^\pi \neq V^{\pi'}$, then $\pi$ and $\pi'$ cannot have the same set of improvable states.

Contradiction!
Basic Tool: Property of Improvement sets in 2-action MDPs

Consider $\pi, \pi' \in \Pi$. If $V^\pi \neq V^{\pi'}$, then $\pi$ and $\pi'$ cannot have the same set of improvable states.

Contradiction!
Basic Tool: Property of Improvement sets in 2-action MDPs

Consider $\pi, \pi' \in \Pi$. If $V^\pi \neq V^{\pi'}$, then $\pi$ and $\pi'$ cannot have the same set of improvable states.

Contradiction!

Equal value functions.
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.
Howard’s Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

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

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

Switch actions in every improvable state.

Possible?

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

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

Switch actions in every improvable state.

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

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

<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>$\pi$</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>

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

Switch actions in *every* improvable state.

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

| $\pi'$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\pi_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| $\pi_2$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| $\pi_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| $\pi$  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

Switch actions in *every* improvable state.

| $\pi'$ | 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

Shivaram Kalyanakrishnan (2017) Theoretical Analysis of Policy Iteration
Switch actions in *every* improvable state.

| \( \pi' \) | 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 | 1 | 0 | 0 |
| \( \pi_3 \) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| \( \pi_4 \) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| \( \pi \) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

If \( \pi \) has *m* improvable states and \( \pi \xrightarrow{\text{Howard’s PI}} \pi' \), then there exist *m* policies \( \pi'' \) such that \( \pi' \succeq \pi'' \succ \pi \).
Howard’s Policy Iteration (2-action MDPs)

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

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].
Randomised Policy Iteration (2-action MDPs)

From the set of improving states, pick a non-empty subset \( S_i \) uniformly at random. Switch actions of all states in \( S_i \).
Randomised Policy Iteration (2-action MDPs)

From the set of improving states, pick a non-empty subset $S_i$ uniformly at random. Switch actions of all states in $S_i$.

\[
\pi = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Randomised Policy Iteration (2-action MDPs)

From the set of improving states, pick a non-empty subset $S_i$ uniformly at random. Switch actions of all states in $S_i$.

<table>
<thead>
<tr>
<th>$\pi$</th>
<th>0 0 0 0 0 0 0 0 0 1 1 1</th>
<th>$\frac{1}{7}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_7$</td>
<td>0 0 0 0 0 0 0 0 0 1 1 1</td>
<td>$\frac{1}{7}$</td>
</tr>
<tr>
<td>$\pi_6$</td>
<td>0 0 0 0 0 0 0 0 0 1 1 0</td>
<td>$\frac{1}{7}$</td>
</tr>
<tr>
<td>$\pi_5$</td>
<td>0 0 0 0 0 0 0 0 0 1 0 1</td>
<td>$\frac{1}{7}$</td>
</tr>
<tr>
<td>$\pi_4$</td>
<td>0 0 0 0 0 0 0 0 0 1 0 0</td>
<td>$\frac{1}{7}$</td>
</tr>
<tr>
<td>$\pi_3$</td>
<td>0 0 0 0 0 0 0 0 0 1 1 1</td>
<td>$\frac{1}{7}$</td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>0 0 0 0 0 0 0 0 0 1 0 0</td>
<td>$\frac{1}{7}$</td>
</tr>
<tr>
<td>$\pi_1$</td>
<td>0 0 0 0 0 0 0 0 0 0 0 1</td>
<td>$\frac{1}{7}$</td>
</tr>
<tr>
<td>$\pi$</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>Probability</td>
</tr>
</tbody>
</table>

Probability
Randomised Policy Iteration (2-action MDPs)

From the set of improving states, pick a non-empty subset $S_i$ uniformly at random.
Switch actions of all states in $S_i$.

If $\pi$ has $m$ improvable states and $\pi \xrightarrow{\text{Randomised PI}} \pi'$, then with probability $1/2$, there exist $2^{m-1}$ policies $\pi''$ such that $\pi'' \succ \pi$ and $\neg (\pi'' \succ \pi')$.
Randomised Policy Iteration (2-action MDPs)

From the set of improving states, pick a non-empty subset $S_i$ uniformly at random. Switch actions of all states in $S_i$.

$$
\begin{align*}
\pi_7 &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1/7 \\
\pi_6 &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1/7 \\
\pi_5 &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1/7 \\
\pi_4 &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1/7 \\
\pi_3 &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1/7 \\
\pi_2 &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1/7 \\
\pi_1 &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1/7 \\
\pi &\quad 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \text{Probability} \\
\end{align*}
$$

If $\pi$ has $m$ improvable states and $\pi \xrightarrow{\text{Randomised PI}} \pi'$, then with probability $1/2$, there exist $2^{m-1}$ policies $\pi''$ such that $\pi'' \succ \pi$ and $\neg(\pi'' \succ \pi')$.

Number of policies eliminated is exponential in $m$. As before, $m^*$ can be tuned such that the expected number of iterations taken by Randomised PI = $O(1.7172^n)$ [MS99].
Batch-Switching Policy Iteration (BSPI)

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

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

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.

\[ \pi_1 \]

\[
\begin{array}{ccccccccc}
S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 & S_{10} \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}
\]
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.

\[
\begin{align*}
\pi_3 &\quad 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\pi_2 &\quad 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\pi_1 &\quad 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{align*}
\]

States: \( S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10} \)
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>

$s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}$
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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</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>
<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>
<tr>
<td></td>
<td></td>
<td></td>
<td></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>0</th>
<th>1</th>
<th>1</th>
<th>0</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>
</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>
</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>
</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.

![Policy Iteration Diagram]

- 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).
- $T(n) \leq 3 \times T(n - 2) \leq \sqrt{3}^n$. 

Shivaram Kalyanakrishnan (2017) Theoretical Analysis of Policy Iteration
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 drawn 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 drawn above are called Trajectory-bounding Trees (TBTs) [KMG16a] (and correspond to the Order Regularity Problem [H12, GHDJ15]).

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

Shivaram Kalyanakrishnan (2017) Theoretical Analysis of Policy Iteration 23 / 31
Batch-Switching Policy Iteration (BSPI)

Principle of constructing TBTs:

\[ L^{+}_{\pi, IS} \overset{\text{def}}{=} \{ \pi' \in \Pi : \exists s \in IS(\pi'(s) \neq \pi(s)) \land \forall s \in (S \setminus IS)(\pi'(s) = \pi(s)) \} ; \]

\[ L^{-}_{\pi, IS} \overset{\text{def}}{=} \{ \pi' \in \Pi : \forall s \in IS(\pi'(s) = \pi(s)) \} . \]
Batch-Switching Policy Iteration (BSPI)

Principle of constructing TBTs:

\[ L^{+}_{\pi, IS} = \{ \pi' \in \Pi : \exists s \in IS(\pi'(s) \neq \pi(s)) \land \forall s \in (S \setminus IS)(\pi'(s) = \pi(s)) \} \];

\[ L^{-}_{\pi, IS} = \{ \pi' \in \Pi : \forall s \in IS(\pi'(s) = \pi(s)) \} \].

<table>
<thead>
<tr>
<th>\pi, IS:</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ L^{-}_{\pi, IS} ]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>[ L^{+}_{\pi, IS} ]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Batch-Switching Policy Iteration (BSPI)

Principle of constructing TBTs:

\[ L^+_\pi,IS \overset{\text{def}}{=} \{ \pi' \in \Pi : \exists s \in IS(\pi'(s) \neq \pi(s)) \land \forall s \in (S \setminus IS)(\pi'(s) = \pi(s)) \} \]

\[ L^-\pi,IS \overset{\text{def}}{=} \{ \pi' \in \Pi : \forall s \in IS(\pi'(s) = \pi(s)) \} \]

<table>
<thead>
<tr>
<th>( L^+_\pi,IS )</th>
<th>1 1 1 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 1 0 0 0</td>
</tr>
<tr>
<td></td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td></td>
<td>1 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>0 1 1 0 0</td>
</tr>
<tr>
<td></td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td></td>
<td>0 0 1 0 0</td>
</tr>
<tr>
<td>( \pi, IS: )</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>( L^-\pi,IS )</td>
<td>0 0 0 0 1</td>
</tr>
<tr>
<td></td>
<td>0 0 0 1 0</td>
</tr>
<tr>
<td></td>
<td>0 0 0 1 1</td>
</tr>
</tbody>
</table>

If \((\pi_1, IS_1), (\pi_2, IS_2), \ldots, (\pi_t, IS_t)\) is a trajectory encountered by PI, it must satisfy, for \(1 \leq i < j \leq t\):

\[ L^-\pi_i,IS_i \cap L^+\pi_j,IS_j = \emptyset. \]
BSPI: 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>
## BSPI: 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].
### BSPI: 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?
**BSPI: 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 a 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].
Overview

1. **Background**
   - MDP Planning
   - Bellman’s Equations and Bellman’s Optimality Equations
   - Solution strategies
   - Strong Running-time Bounds

2. **Policy Iteration**
   - Policy Improvement
   - Proof of Policy Improvement Theorem
   - Policy Iteration algorithm
   - Switching strategies and bounds

3. **Analysis of Policy Iteration on 2-action MDPs**
   - Basic Tools and Results
   - Howard’s Policy Iteration
   - Mansour and Singh’s Randomised Policy Iteration
   - Batch-Switching Policy Iteration

4. **Summary and Outlook**
   - Results for \( k \)-action MDPs
   - Open problems
   - References
   - Conclusion
Policy Iteration on $k$-action MDPs

- What are the main differences between 2-action and $k$-action MDPs ($k > 2$)?
What are the main differences between 2-action and $k$-action MDPs ($k > 2$)?

- In $k$-action MDPs, states can be both improvable and deprovable.
- In $k$-action MDPs, there can be more than one improving action.
Policy Iteration on \( k \)-action MDPs

- What are the main differences between 2-action and \( k \)-action MDPs \((k > 2)\)?
  
  In \( k \)-action MDPs, states can be both improvable and deprovable. In \( k \)-action MDPs, there can be more than one improving action.

- Mansour and Singh’s analysis makes no assumption on which improving action is picked, only that one is picked at all, in the states selected to be switched.

  Bound for Howard’s PI: \( O \left( \frac{k^n}{n} \right) \) iterations [MS99, HGDJ14].

  Bound for Randomised PI: \( O \left( \left( \left( 1 + \frac{2}{\log(k)} \right)^{\frac{k}{2}} \right)^n \right) \) expected iterations [MS99].
Policy Iteration on $k$-action MDPs

- What are the main differences between 2-action and $k$-action MDPs ($k > 2$)?
  
  In $k$-action MDPs, states can be both improvable and deprovable. In $k$-action MDPs, there can be more than one improving action.

- Mansour and Singh’s analysis makes no assumption on which improving action is picked, only that one is picked at all, in the states selected to be switched.
  
  Bound for Howard’s PI: $O\left(\frac{k^n}{n}\right)$ iterations [MS99, HGDJ14].
  
  Bound for Randomised PI: $O\left(\left(\left(1 + \frac{2}{\log(k)}\right) \frac{k}{2}\right)^n\right)$ expected iterations [MS99].

- Randomised Simple PI [KMG16b]: Switch only the “rightmost” improvable state; switch to an improving action picked uniformly at random.
  
  Bound: $(2 + \ln(k - 1))^n$ expected iterations.
Policy Iteration on $k$-action MDPs

- **What are the main differences between 2-action and $k$-action MDPs ($k > 2$)?**
  - In $k$-action MDPs, states can be both improvable and deprovable.
  - In $k$-action MDPs, there can be more than one improving action.

- Mansour and Singh’s analysis makes **no assumption** on which improving action is picked, only that one is picked at all, in the states selected to be switched.
  - Bound for Howard’s PI: $O\left(\frac{k^n}{n}\right)$ iterations [MS99, HGDJ14].
  - Bound for Randomised PI: $O\left(\left(\left(1 + \frac{2}{\log(k)}\right)\frac{k}{2}\right)^n\right)$ expected iterations [MS99].

- **Randomised Simple PI** [KMG16b]: Switch only the “rightmost” improvable state; switch to an improving action picked **uniformly at random**.
  - Bound: $(2 + \ln(k - 1))^n$ expected iterations.

- **Recursive BSPI** [GK17]: Deterministic switching strategy based on a **binary hierarchy** of actions (that facilitates reusing the 2-action MDP analysis).
  - Bound: $k^{0.7207n}$ iterations.
Open Problems

- Is the complexity of Howard’s PI on 2-action MDPs upper-bounded by the Fibonacci sequence ($\approx 1.6181^n$)?
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)?
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 iterations taken by Howard’s PI on 2-action MDPs?
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 iterations taken by Howard’s PI on 2-action MDPs?

- Is (Howard’s) PI strongly polynomial on deterministic MDPs?
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 iterations taken by Howard’s PI on 2-action MDPs?

- Is (Howard’s) PI strongly polynomial on deterministic MDPs?

- Does PI admit a smoothed analysis similar to the Simplex algorithm for Linear Programming [ST04]?
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 iterations taken by Howard’s PI on 2-action MDPs?

- Is (Howard’s) PI strongly polynomial on deterministic MDPs?

- Does PI admit a smoothed analysis similar to the Simplex algorithm for Linear Programming [ST04]?

- Is there a strongly polynomial algorithm for MDP planning?
References


References


Conclusion

- Policy Iteration is an elegant family of algorithms for MDP Planning.

- Under the infinite precision arithmetic computation model, it naturally yields strong running time bounds, which depend only on the number of states and actions.

- This tutorial is prompted by some recent progress that has resulted in exponential improvements in upper bounds.

- The main tool of analysis remains basic: the well-known Policy Improvement Theorem.

- Both theory and experiments suggest that Howard’s Policy Iteration could be more efficient than it has formally been proven.

- The vast gap between the upper and lower bounds motivates several interesting questions for future research.
Conclusion

- Policy Iteration is an elegant family of algorithms for MDP Planning.

- Under the infinite precision arithmetic computation model, it naturally yields strong running time bounds, which depend only on the number of states and actions.

- This tutorial is prompted by some recent progress that has resulted in exponential improvements in upper bounds.

- The main tool of analysis remains basic: the well-known Policy Improvement Theorem.

- Both theory and experiments suggest that Howard’s Policy Iteration could be more efficient than it has formally been proven.

- The vast gap between the upper and lower bounds motivates several interesting questions for future research.

Thank you!