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

Overview

MDP Planning

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

- An MDP comprises a tuple (S, A, R, T, γ), 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 $\pi : S \to A$ 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 π from state s is:

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

$\gamma \in [0, 1)$ is a discount factor.

MDP Planning

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

- An MDP comprises a tuple (S, A, R, T, γ), 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 π from state s is:

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

$\gamma \in [0, 1)$ is a discount factor.

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

MDP Planning

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

- An MDP comprises a tuple (S, A, R, T, γ), 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 $\pi : S \to A$ 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 π from state s is:

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

$\gamma \in [0, 1)$ is a discount factor.

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

MDP Planning

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

- An MDP comprises a tuple (S, A, R, T, γ), where
  - S is a set of states (with $|S| = n$),
  - A is a set of actions (with $|A| = k$), $\longleftarrow$ (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 $\pi : S \to A$ specifies an action from each state, and yields trajectory

$$s^0, a^0 = \pi(s^0), r^0, s^1, a^1 = \pi(s^1), r^1, s^2, \ldots.$$

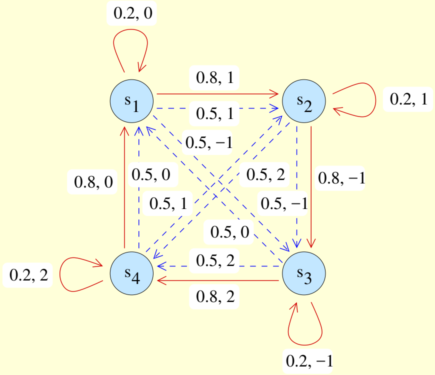
- The value of a policy π from state s is:

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

$\gamma \in [0, 1)$ is a discount factor.

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

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

Discount factor: $\gamma = 0.9$.

Bellman's Equations

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

  Bellman's Equations: $\forall s \in S$,

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

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

Bellman's Equations

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

  Bellman's Equations: $\forall s \in S$,

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

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

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

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

  Q^π is called the action value function of π.

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

Bellman's Equations

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

  Bellman's Equations: $\forall s \in S$,

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

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

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

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

  Q^π is called the action value function of π.

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

- The variables in Bellman's Equations are the elements of V^π.

  n linear equations in n unknowns.

Bellman's Equations

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

  Bellman's Equations: $\forall s \in S$,

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

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

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

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

  Q^π is called the action value function of π.

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

- The variables in Bellman's Equations are the elements of V^π.

  n linear equations in n unknowns.

> Given S, A, T, R, γ, and a fixed policy π, we can solve Bellman's Equations to obtain V^π and Q^π. This step is called Policy Evaluation.

Bellman's Optimality Equations

- The Optimal Value Function $V^\star \stackrel{\text{def}}{=} V^{\pi^\star}$ is unique solution of: $\forall s \in S$,

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

These are Bellman's Optimality Equations.

- The Optimal Action Value Function $Q^\star \stackrel{\text{def}}{=} Q^{\pi^\star}$ is given by: $\forall s \in S, \forall a \in A$,

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

- Given $Q^\star$, we may obtain $\pi^\star$ by setting, $\forall s \in S$:

$$\boxed{\pi^\star(s) \leftarrow \operatorname{argmax}_{a \in A} Q^\star(s,a).}$$

Given $\pi^\star$, how can we obtain $V^\star$ and $Q^\star$?

Bellman's Optimality Equations

- The Optimal Value Function $V^{\star} \stackrel{\text{def}}{=} V^{\pi^{\star}}$ is unique solution of: $\forall s \in S$,

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

These are Bellman's Optimality Equations.

- The Optimal Action Value Function $Q^{\star} \stackrel{\text{def}}{=} Q^{\pi^{\star}}$ is given by: $\forall s \in S, \forall a \in A$,

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

- Given $Q^{\star}$, we may obtain $\pi^{\star}$ by setting, $\forall s \in S$:

$$\boxed{\pi^{\star}(s) \leftarrow \text{argmax}_{a \in A} Q^{\star}(s, a).}$$

Given $\pi^{\star}$, how can we obtain $V^{\star}$ and $Q^{\star}$? By policy evaluation (previous slide).

Solution Strategies

- Value Iteration

$V_0 \leftarrow$ Arbitrary, element-wise bounded, n-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 $V^\star$ guaranteed using a max-norm contraction argument.

- Linear Programming

$$\text{Minimise} \quad \sum_{s \in S} V(s)$$

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

n variables, nk constraints (or *dual* with nk variables, n 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]:
  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]:
  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]:
  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, γ, etc.).

Strong Running-time Bounds

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

- Upper Bound for Value Iteration [LDK95]:
  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, γ, 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]:
  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, γ, etc.).
  Is there a strong upper bound on the complexity of *policy evaluation*? $O(n^2k + n^3)$.

Strong Running-time Bounds

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

- Upper Bound for Value Iteration [LDK95]:
  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, γ, etc.).
  Is there a strong upper bound on the complexity of *policy evaluation*? $O(n^2k + n^3)$.
  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]:
  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, γ, etc.).
  Is there a strong upper bound on the complexity of *policy evaluation*? $O(n^2k + n^3)$.
  Can you give a strong bound on the running time of MDP planning? 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]:
  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, γ, etc.).
  Is there a strong upper bound on the complexity of *policy evaluation*? $O(n^2k + n^3)$.
  Can you give a strong bound on the running time of MDP planning? poly$(n, k) \cdot k^n$.

- Bounds for Linear Programming-type approaches to MDP planning:
  poly(n, k, B) [K80, K84].
  poly$(n, k) \cdot \exp(O(\sqrt{n \log(n)}))$ (Expected) [MSW96].
  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]:
  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, γ, etc.).
  Is there a strong upper bound on the complexity of *policy evaluation*? $O(n^2k + n^3)$.
  Can you give a strong bound on the running time of MDP planning? poly$(n, k) \cdot k^n$.

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

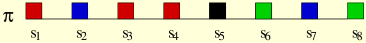
- Strong bounds depend solely on n and k (no dependence on B, γ, etc.).
  Is there a strong upper bound on the complexity of *policy evaluation*? $O(n^2k + n^3)$.
  Can you give a strong bound on the running time of MDP planning? $\text{poly}(n, k) \cdot k^n$.

- 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].

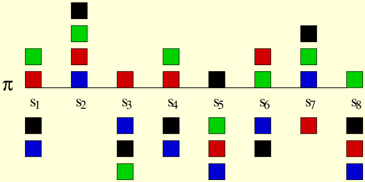
- Appeal of Policy Iteration:
  Theoretical: naturally yields strong bounds (also enjoys good weak bounds [P94]).
  Practical: very fast on MDPs encountered in typical applications.

Overview

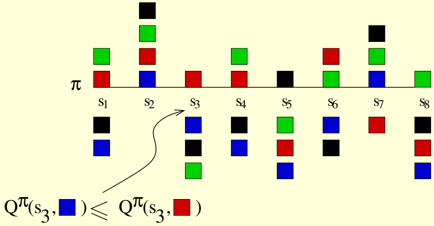
Policy Improvement



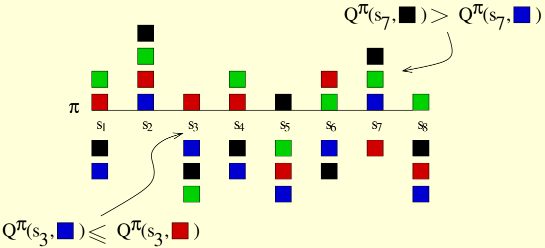
Policy Improvement



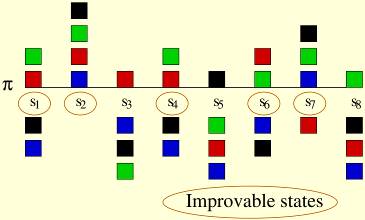
Policy Improvement



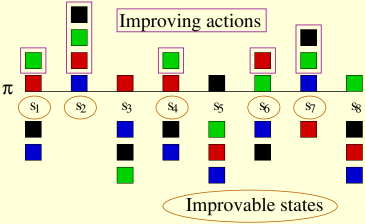
Policy Improvement



Policy Improvement

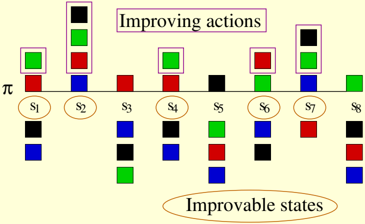


Policy Improvement



Policy Improvement

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



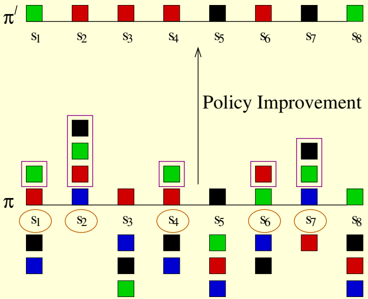
Policy Improvement

Given π,

Pick one or more improvable states, and in them,

Switch to an arbitrary improving action.

Let the resulting policy be π'.



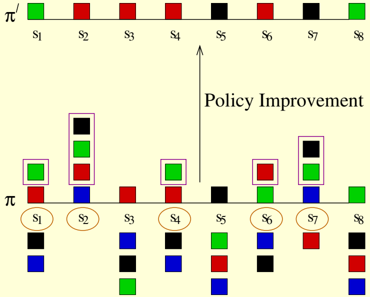
Policy Improvement

Given π,

Pick one or more improvable states, and in them,

Switch to an arbitrary improving action.

Let the resulting policy be π'.



Policy Improvement Theorem:

(1) If π has no improvable states, then it is optimal, else

(2) if π' is obtained as above, then

$$\forall s \in S : V^{\pi'}(s) \geq V^{\pi}(s) \text{ and } \exists s \in S : V^{\pi'}(s) > V^{\pi}(s).$$

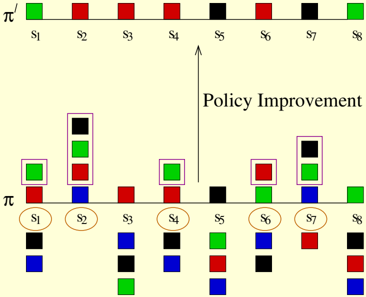
Policy Improvement

Given π,

  Pick one or more improvable states, and in them,

  Switch to an arbitrary improving action.

Let the resulting policy be π'.



Policy Improvement Theorem:

(1) If π has no improvable states, then it is optimal, else

(2) if π' 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 $X : S \to \mathbb{R}$ 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 $\exists s \in S : X(s) > Y(s)$.

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 $V^{\pi_1} \succ V^{\pi_2}$.

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 $\forall s \in S$,

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

Definitions and Basic Facts

- For $X : S \to \mathbb{R}$ 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 $\forall s \in S$,

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

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

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

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 $\forall s \in S$,

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

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

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

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

π has no improvable states

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

Proof of Policy Improvement Theorem

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

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

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

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

π has no improvable states

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

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

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

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

π has improvable states and policy improvement yields π'

Proof of Policy Improvement Theorem

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

π has no improvable states

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

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

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

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

π has improvable states and policy improvement yields π'

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

Proof of Policy Improvement Theorem

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

π has no improvable states

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

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

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

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

π has improvable states and policy improvement yields π'

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

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

Proof of Policy Improvement Theorem

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

π has no improvable states

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

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

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

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

π has improvable states and policy improvement yields π'

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

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

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

Proof of Policy Improvement Theorem

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

π has no improvable states

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

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

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

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

π has improvable states and policy improvement yields π'

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

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

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

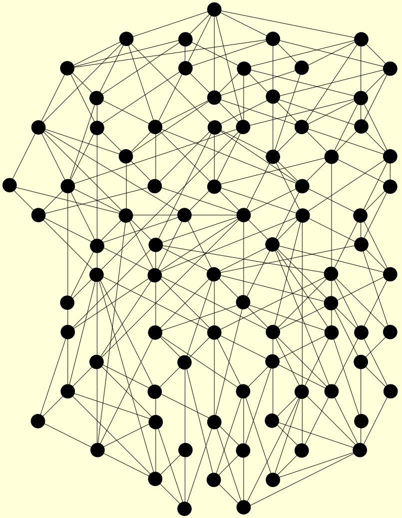
$$\implies V^{\pi'} \succ V^\pi.$$

Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.
While π has improvable states:
    $\pi \leftarrow$ PolicyImprovement(π).

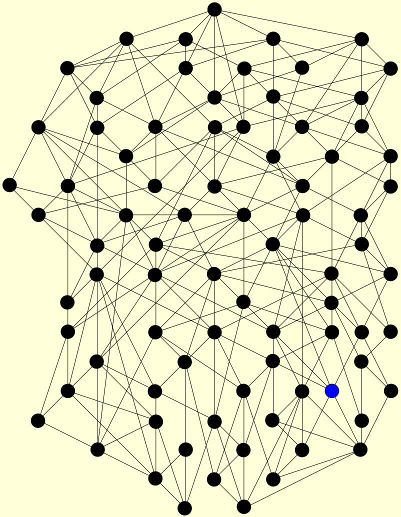
Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.
While π has improvable states:
    $\pi \leftarrow$ PolicyImprovement(π).



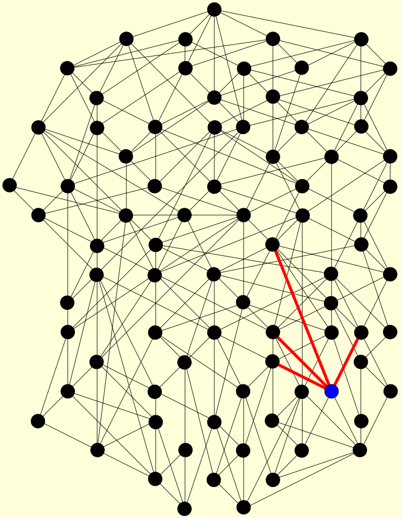
Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.
While π has improvable states:
    $\pi \leftarrow$ PolicyImprovement(π).



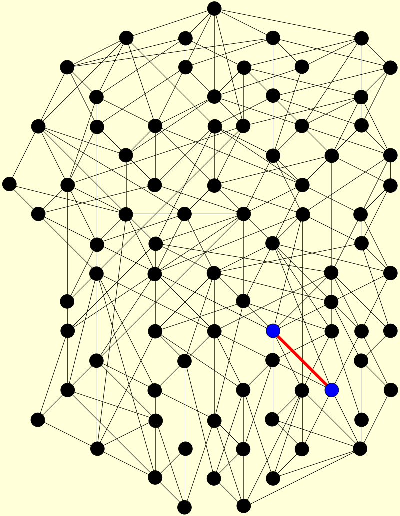
Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.
While π has improvable states:
    $\pi \leftarrow$ PolicyImprovement(π).



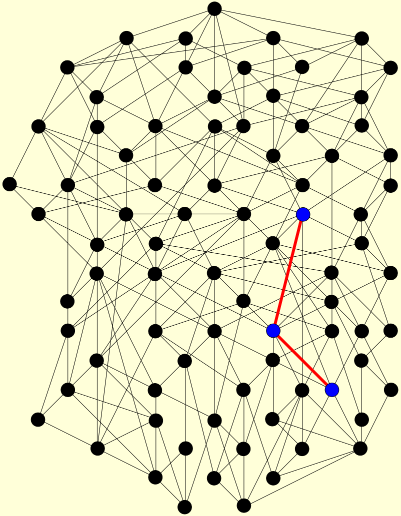
Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.
While π has improvable states:
    $\pi \leftarrow$ PolicyImprovement(π).



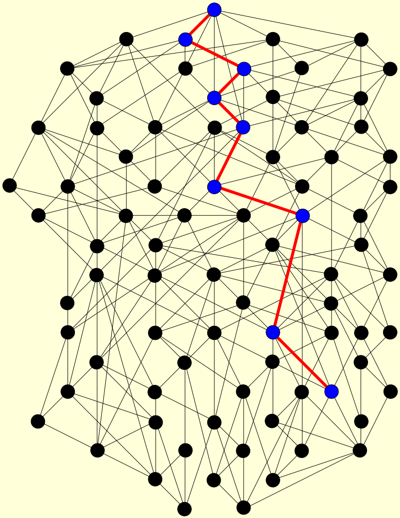
Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.
While π has improvable states:
$\pi \leftarrow$ PolicyImprovement(π).



Policy Iteration Algorithm

$\pi \leftarrow$ Arbitrary policy.
While π has improvable states:
    $\pi \leftarrow$ PolicyImprovement(π).



Switching Strategies and Bounds

Upper bounds on number of iterations

PI Variant	**Type**	$k = 2$	**General** k
Howard's PI [H60, MS99]	Deterministic	$O\left(\frac{2^n}{n}\right)$	$O\left(\frac{k^n}{n}\right)$
Mansour and Singh's Randomised PI [MS99]	Randomised	1.7172^n	$\approx O\left(\left(\frac{k}{2}\right)^n\right)$

Switching Strategies and Bounds

Upper bounds on number of iterations

PI Variant	**Type**	$k = 2$	**General** k
Howard's PI [H60, MS99]	Deterministic	$O\left(\frac{2^n}{n}\right)$	$O\left(\frac{k^n}{n}\right)$
Mansour and Singh's Randomised PI [MS99]	Randomised	1.7172^n	$\approx O\left(\left(\frac{k}{2}\right)^n\right)$
Batch-switching PI (BSPI) [KMG16a]	Deterministic	1.6479^n	–
Recursive BSPI [GK17]	Deterministic	–	$k^{0.7207n}$
Recursive Simple PI [KMG16b]	Randomised	–	$(2 + \ln(k-1))^n$

Switching Strategies and Bounds

Upper bounds on number of iterations

PI Variant	**Type**	$k = 2$	**General** k
Howard's PI [H60, MS99]	Deterministic	$O\left(\frac{2^n}{n}\right)$	$O\left(\frac{k^n}{n}\right)$
Mansour and Singh's Randomised PI [MS99]	Randomised	1.7172^n	$\approx O\left(\left(\frac{k}{2}\right)^n\right)$
Batch-switching PI (BSPI) [KMG16a]	Deterministic	1.6479^n	–
Recursive BSPI [GK17]	Deterministic	–	$k^{0.7207n}$
Recursive Simple PI [KMG16b]	Randomised	–	$(2 + \ln(k-1))^n$

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].

Switching Strategies and Bounds

Upper bounds on number of iterations

PI Variant	**Type**	$k = 2$	**General** k
Howard's PI [H60, MS99]	Deterministic	$O\left(\frac{2^n}{n}\right)$	$O\left(\frac{k^n}{n}\right)$
Mansour and Singh's Randomised PI [MS99]	Randomised	1.7172^n	$\approx O\left(\left(\frac{k}{2}\right)^n\right)$
Batch-switching PI (BSPI) [KMG16a]	Deterministic	1.6479^n	–
Recursive BSPI [GK17]	Deterministic	–	$k^{0.7207n}$
Recursive Simple PI [KMG16b]	Randomised	–	$(2 + \ln(k-1))^n$

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

PI Variant	**Type**	$k = 2$	**General** k
Howard's PI [H60, MS99]	Deterministic	$O\left(\frac{2^n}{n}\right)$	$O\left(\frac{k^n}{n}\right)$
Mansour and Singh's Randomised PI [MS99]	Randomised	1.7172^n	$\approx O\left(\left(\frac{k}{2}\right)^n\right)$
Batch-switching PI (BSPI) [KMG16a]	Deterministic	1.6479^n	–
Recursive BSPI [GK17]	Deterministic	–	$k^{0.7207n}$
Recursive Simple PI [KMG16b]	Randomised	–	$(2 + \ln(k-1))^n$

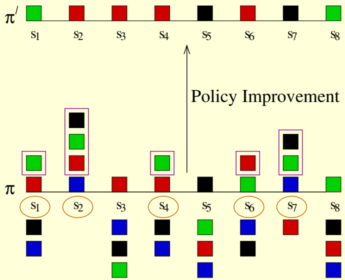
Lower bounds on number of iterations

$\Omega(2^{n/7})$ 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].
$\Omega(n)$ Howard's PI on n-state, 2-action MDPs [HZ10].

Overview

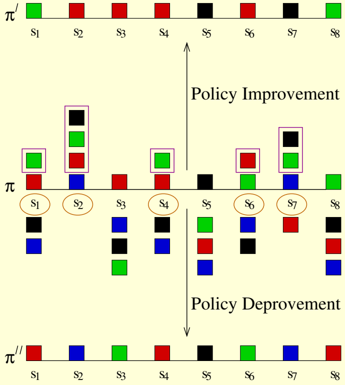
Basic Tool: Policy Improvement and Policy Deprovement

$$\boxed{\pi' \succ \pi.}$$



Basic Tool: Policy Improvement and Policy Deprovement

$$\boxed{\pi' \succ \pi.}$$



$$\boxed{\pi \succeq \pi''.}$$

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

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

$\boxed{1}$ $\boxed{0}$ $\boxed{1}$ $\boxed{1}$ 0 1 1 0 $\quad$ $\boxed{1}$ $\boxed{1}$ $\boxed{0}$ $\boxed{0}$ 1 1 0 1

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

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

1 1 0 0 0 1 1 0

$\succ$

$\boxed{\mathbf{1}}$ $\boxed{\mathbf{0}}$ $\boxed{\mathbf{1}}$ $\boxed{\mathbf{1}}$ **0 1 1 0** $\boxed{\mathbf{1}}$ $\boxed{\mathbf{1}}$ $\boxed{\mathbf{0}}$ $\boxed{\mathbf{0}}$ **1 1 0 1**

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

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

$$\mathbf{1} \quad \mathbf{1} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{0}$$

$$\succ$$

$$\boxed{\mathbf{1}} \quad \boxed{\mathbf{0}} \quad \boxed{\mathbf{1}} \quad \boxed{\mathbf{1}} \quad \mathbf{0} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{0} \qquad \boxed{\mathbf{1}} \quad \boxed{\mathbf{1}} \quad \boxed{\mathbf{0}} \quad \boxed{\mathbf{0}} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{0} \quad \mathbf{1}$$

$$\succeq$$

$$\mathbf{1} \quad \mathbf{0} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{0} \quad \mathbf{1}$$

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

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

$$1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0$$
$$\succ$$
$$\boxed{1} \quad \boxed{0} \quad \boxed{1} \quad \boxed{1} \quad 0 \quad 1 \quad 1 \quad 0$$
$$\succeq$$
$$1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1$$

$$1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1$$
$$\succ$$
$$\boxed{1} \quad \boxed{1} \quad \boxed{0} \quad \boxed{0} \quad 1 \quad 1 \quad 0 \quad 1$$

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

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

$$\begin{array}{cccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
 & & & \succ & & & & \\
\boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} & 0 & 1 & 1 & 0 \\
 & & & \succeq & & & & \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\qquad
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & & & \succ & & & & \\
\boxed{1} & \boxed{1} & \boxed{0} & \boxed{0} & 1 & 1 & 0 & 1 \\
 & & & \succeq & & & & \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}$$

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

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

$$\begin{array}{cccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
 & & & \succ & & & & \\
\boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} & 0 & 1 & 1 & 0 \\
 & & & \succeq & & & & \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\qquad
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & & & \succ & & & & \\
\boxed{1} & \boxed{1} & \boxed{0} & \boxed{0} & 1 & 1 & 0 & 1 \\
 & & & \succeq & & & & \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}$$

Contradiction!

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

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

1 1 0 0 0 1 1 0   **1 0 1 1 1 1 0 1**

$\succ$ $\succ$

$\boxed{1}$ $\boxed{0}$ $\boxed{1}$ $\boxed{1}$ **0 1 1 0** $\boxed{1}$ $\boxed{1}$ $\boxed{0}$ $\boxed{0}$ **1 1 0 1**

$\succeq$ $\succeq$

1 0 1 1 1 1 0 1   **1 1 0 0 0 1 1 0**

Contradiction!

$\boxed{1}$ $\boxed{0}$ $\boxed{1}$ $\boxed{1}$ **0 1 1 0** $\boxed{1}$ $\boxed{0}$ $\boxed{1}$ $\boxed{1}$ **1 1 0 1**

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

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

$$
\begin{array}{cccccccc}
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
 & & & \succ & & & & \\
\boxed{1} & \boxed{0} & \boxed{1} & \boxed{1} & 0 & 1 & 1 & 0 \\
 & & & \succeq & & & & \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
\qquad
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & & & \succ & & & & \\
\boxed{1} & \boxed{1} & \boxed{0} & \boxed{0} & 1 & 1 & 0 & 1 \\
 & & & \succeq & & & & \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}
$$

Contradiction!

$$
\boxed{1}\ \boxed{0}\ \boxed{1}\ \boxed{1}\ 0\ 1\ 1\ 0 \qquad \boxed{1}\ \boxed{0}\ \boxed{1}\ \boxed{1}\ 1\ 1\ 0\ 1
$$

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.

π **0 0 0 0 0 0 0** $\boxed{\mathbf{0}}$ $\boxed{\mathbf{0}}$ $\boxed{\mathbf{0}}$ $\boxed{\mathbf{0}}$ $\boxed{\mathbf{0}}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

π'	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**
π	**0**	**0**	**0**	**0**	**0**	**0**	**0**	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

Possible?

π'	**0**	$\boxed{\mathbf{0}}$	**0**	**0**	$\boxed{\mathbf{0}}$	**0**	**0**	$\boxed{\mathbf{1}}$	$\boxed{\mathbf{1}}$	$\boxed{\mathbf{1}}$	$\boxed{\mathbf{1}}$	$\boxed{\mathbf{1}}$
π	**0**	**0**	**0**	**0**	**0**	**0**	**0**	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

π'	**0**	$\boxed{0}$	**0**	**0**	$\boxed{0}$	**0**	**0**	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	**1**
π	**0**	**0**	**0**	**0**	**0**	**0**	**0**	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

π'	0	$\boxed{0}$	0	0	$\boxed{0}$	0	0	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	1
π_1	0	0	0	0	0	0	0	1	1	1	1	0
π	0	0	0	0	0	0	0	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

π'	**0**	$\boxed{0}$	**0**	**0**	$\boxed{0}$	**0**	**0**	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	**1**
π_1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**
π_2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**
π	**0**	**0**	**0**	**0**	**0**	**0**	**0**	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

π'	0	$\boxed{0}$	0	0	$\boxed{0}$	0	0	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	1
π_1	0	0	0	0	0	0	0	1	1	1	1	0
π_2	0	0	0	0	0	0	0	1	1	1	0	0
π_3	0	0	0	0	0	0	0	1	1	0	0	0
π	0	0	0	0	0	0	0	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

π'	0	$\boxed{0}$	0	0	$\boxed{0}$	0	0	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	1
π_1	0	0	0	0	0	0	0	1	1	1	1	0
π_2	0	0	0	0	0	0	0	1	1	1	0	0
π_3	0	0	0	0	0	0	0	1	1	0	0	0
π_4	0	0	0	0	0	0	0	1	0	0	0	0
π	0	0	0	0	0	0	0	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

Howard's Policy Iteration (2-action MDPs)

Switch actions in every improvable state.

π'	0	$\boxed{0}$	0	0	$\boxed{0}$	0	0	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$	1
π_1	0	0	0	0	0	0	0	1	1	1	1	0
π_2	0	0	0	0	0	0	0	1	1	1	0	0
π_3	0	0	0	0	0	0	0	1	1	0	0	0
π_4	0	0	0	0	0	0	0	1	0	0	0	0
π	0	0	0	0	0	0	0	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$

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

Howard's Policy Iteration (2-action MDPs)

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

Howard's Policy Iteration (2-action MDPs)

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

Howard's Policy Iteration (2-action MDPs)

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

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

Howard's Policy Iteration (2-action MDPs)

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

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

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

Howard's Policy Iteration (2-action MDPs)

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

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

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

$$\leq \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{m^\star - 1}$$

Howard's Policy Iteration (2-action MDPs)

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

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

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

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

Howard's Policy Iteration (2-action MDPs)

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

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

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

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

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.

π **0 0 0 0 0 0 0 0 0** $\boxed{\mathbf{0}}$ $\boxed{\mathbf{0}}$ $\boxed{\mathbf{0}}$

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 .

π_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	$1/7$
π_6	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	$1/7$
π_5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	$1/7$
π_4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	$1/7$
π_3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	$1/7$
π_2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	$1/7$
π_1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	$1/7$
π	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	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 .

π_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	$1/7$
π_6	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	$1/7$
π_5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	$1/7$
π_4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	$1/7$
π_3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	$1/7$
π_2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	$1/7$
π_1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	$1/7$
π	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	Probability

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

π_7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	$1/7$
π_6	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	$1/7$
π_5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	$1/7$
π_4	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	$1/7$
π_3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	$1/7$
π_2	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	$1/7$
π_1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	$1/7$
π	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	Probability

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

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

Batch-Switching Policy Iteration (BSPI)

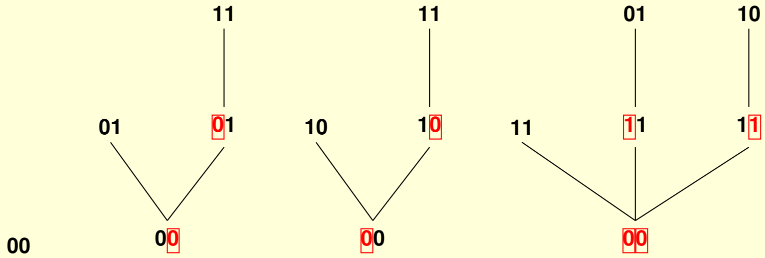
Howard's Policy Iteration takes at most ___ iterations on a 2-state MDP!

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.

$$\pi_1 \quad \left\| \begin{array}{cc} \mathbf{0} & \boxed{\mathbf{1}} \\ s_1 & s_2 \end{array} \right\| \begin{array}{cc} \mathbf{1} & \mathbf{0} \\ s_3 & s_4 \end{array} \left\| \begin{array}{cc} \boxed{\mathbf{0}} & \boxed{\mathbf{0}} \\ s_5 & s_6 \end{array} \right\| \begin{array}{cc} \mathbf{1} & \boxed{\mathbf{0}} \\ s_7 & s_8 \end{array} \left\| \begin{array}{cc} \boxed{\mathbf{0}} & \mathbf{0} \\ s_9 & s_{10} \end{array} \right\|$$

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.

π_2	**0**	$\boxed{\mathbf{1}}$	**1**	**0**	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	**1**	**0**	**1**	**0**
π_1	**0**	$\boxed{\mathbf{1}}$	**1**	**0**	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	**1**	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	**0**
	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.

π_3	$\boxed{0}$	$\boxed{1}$	1	0	1	$\boxed{1}$	1	$\boxed{0}$	1	0
π_2	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	0	1	0
π_1	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	$\boxed{0}$	$\boxed{0}$	0
	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.

π_4	$\boxed{0}$	1	1	0	1	1	1	1	1	0
π_3	$\boxed{0}$	$\boxed{1}$	1	0	1	$\boxed{1}$	1	$\boxed{0}$	1	0
π_2	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	0	1	0
π_1	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	$\boxed{0}$	$\boxed{0}$	0
	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.

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
π_4	$\boxed{0}$	1	1	0	1	1	1	1	1	0
π_3	$\boxed{0}$	$\boxed{1}$	1	0	1	$\boxed{1}$	1	$\boxed{0}$	1	0
π_2	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	0	1	0
π_1	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	$\boxed{0}$	$\boxed{0}$	0

- 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.

π_4	$\boxed{0}$	1	1	0	1	1	1	1	1	0
π_3	$\boxed{0}$	$\boxed{1}$	1	0	1	$\boxed{1}$	1	$\boxed{0}$	1	0
π_2	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	0	1	0
π_1	0	$\boxed{1}$	1	0	$\boxed{0}$	$\boxed{0}$	1	$\boxed{0}$	$\boxed{0}$	0
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}

- 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.

π_4	0	1	1	0	1	1	1	1	1	0
π_3	0	1	1	0	1	1	1	0	1	0
π_2	0	1	1	0	0	0	1	0	1	0
π_1	0	1	1	0	0	0	1	0	0	0
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}

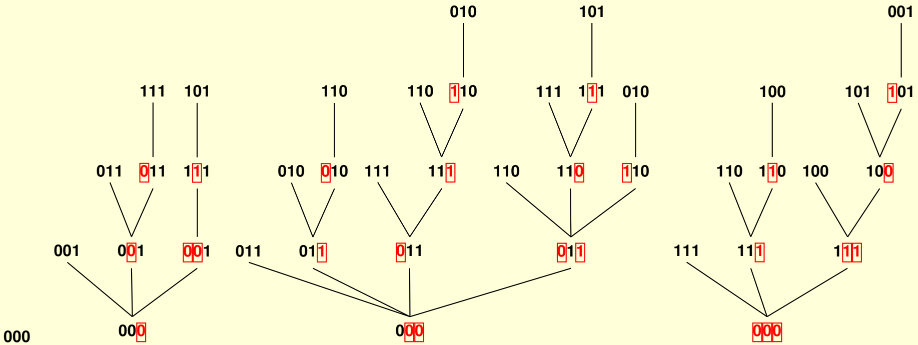
- Left-most batch can change only when all other columns are non-improvable.
- Left-most batch can change at most 3 times (following previous result).
- $T(n) \leq 3 \times T(n-2) \leq \sqrt{3}^n$.

Batch-Switching Policy Iteration (BSPI)

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

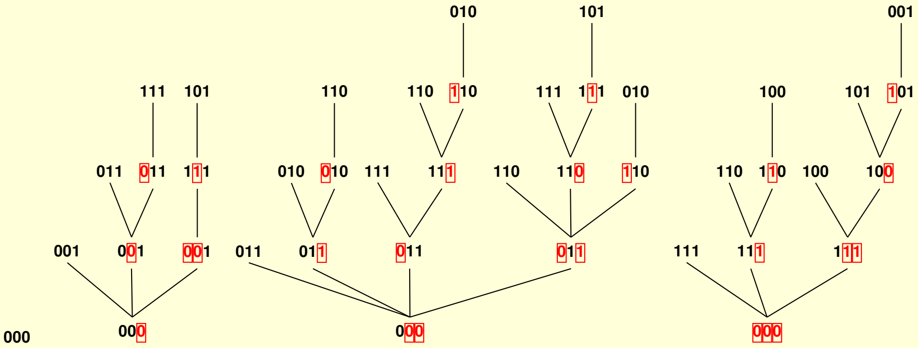
Batch-Switching Policy Iteration (BSPI)

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



Batch-Switching Policy Iteration (BSPI)

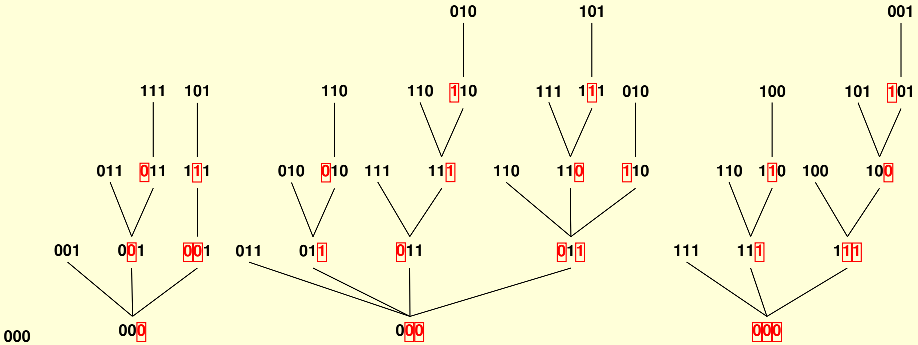
Howard's Policy Iteration takes at most 5 iterations on a 3-state MDP!



The structures 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 $T(n) \leq 5 \times T(n-3) \leq 1.71^n$.

Batch-Switching Policy Iteration (BSPI)

Principle of constructing TBTs:

$L^{+}_{\pi,IS} \stackrel{\text{def}}{=} \{\pi' \in \Pi : \exists s \in IS(\pi'(s) \neq \pi(s)) \wedge \forall s \in (S \setminus IS)(\pi'(s) = \pi(s))\}$;

$L^{-}_{\pi,IS} \stackrel{\text{def}}{=} \{\pi' \in \Pi : \forall s \in IS(\pi'(s) = \pi(s))\}$.

Batch-Switching Policy Iteration (BSPI)

Principle of constructing TBTs:

$L^{+}_{\pi,IS} \stackrel{\text{def}}{=} \{\pi' \in \Pi : \exists s \in IS(\pi'(s) \neq \pi(s)) \wedge \forall s \in (S \setminus IS)(\pi'(s) = \pi(s))\}$;

$L^{-}_{\pi,IS} \stackrel{\text{def}}{=} \{\pi' \in \Pi : \forall s \in IS(\pi'(s) = \pi(s))\}$.

$L^{+}_{\pi,IS}$		**1**	**1**	**1**	**0**	**0**
		1	**1**	**0**	**0**	**0**
		1	**0**	**1**	**0**	**0**
		1	**0**	**0**	**0**	**0**
		0	**1**	**1**	**0**	**0**
		0	**1**	**0**	**0**	**0**
		0	**0**	**1**	**0**	**0**
$L^{-}_{\pi,IS}$	π, IS:	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	**0**	**0**
		0	**0**	**0**	**0**	**1**
		0	**0**	**0**	**1**	**0**
		0	**0**	**0**	**1**	**1**

Batch-Switching Policy Iteration (BSPI)

Principle of constructing TBTs:

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

$L^{-}_{\pi,IS} \stackrel{\text{def}}{=} \{\pi' \in \Pi : \forall s \in IS(\pi'(s) = \pi(s))\}.$

$L^{+}_{\pi,IS}$		**1**	**1**	**1**	**0**	**0**
		1	**1**	**0**	**0**	**0**
		1	**0**	**1**	**0**	**0**
		1	**0**	**0**	**0**	**0**
		0	**1**	**1**	**0**	**0**
		0	**1**	**0**	**0**	**0**
		0	**0**	**1**	**0**	**0**
$L^{-}_{\pi,IS}$	π, IS:	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	$\boxed{\mathbf{0}}$	**0**	**0**
		0	**0**	**0**	**0**	**1**
		0	**0**	**0**	**1**	**0**
		0	**0**	**0**	**1**	**1**

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

Batch size	Depth of TBT	Bound on number of iterations
1	2	2^n
2	3	1.7321^n
3	5	1.7100^n
4	8	1.6818^n
5	13	1.6703^n
6	21	1.6611^n
7	33	1.6479^n

BSPI: Bounds

Batch size	Depth of TBT	Bound on number of iterations
1	2	2^n
2	3	1.7321^n
3	5	1.7100^n
4	8	1.6818^n
5	13	1.6703^n
6	21	1.6611^n
7	33	1.6479^n

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

BSPI: Bounds

Batch size	Depth of TBT	Bound on number of iterations
1	2	2^n
2	3	1.7321^n
3	5	1.7100^n
4	8	1.6818^n
5	13	1.6703^n
6	21	1.6611^n
7	33	1.6479^n

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

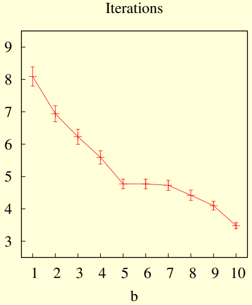
Batch size	Depth of TBT	Bound on number of iterations
1	2	2^n
2	3	1.7321^n
3	5	1.7100^n
4	8	1.6818^n
5	13	1.6703^n
6	21	1.6611^n
7	33	1.6479^n

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

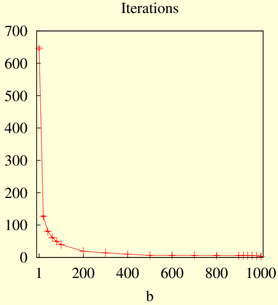
Will the bound continue to be non-increasing in the batch size?

If so, 1.6479^n would be a bound for Howard's Policy Iteration!

BSPI: Effect of Batch Size b



$n = 10$



$n = 1000$

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

Policy Iteration on k-action MDPs

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

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.

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

R. A. Howard, 1960. Dynamic Programming and Markov Processes. MIT Press, 1960.

L. G. Khachiyan, 1980. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72.

N. Karmarkar, 1984. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

Mary Melekopoglou and Anne Condon, 1994. On the complexity of the policy improvement algorithm for Markov decision processes. *INFORMS Journal on Computing*, 6(2):188–192, 1994.

Martin L. Puterman, 1994. Markov Decision Processes. Wiley, 1994.

Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling, 1995. On the complexity of solving Markov decision problems. *In Proc. UAI 1995*, pp. 394–402, Morgan Kaufmann, 1995.

Jiří Matoušek, Micha Sharir, and Emo Welzl, 1996. A Subexponential Bound for Linear Programming. *Algorithmica*, 16(4/5):498–516, 1996.

Yishay Mansour and Satinder Singh, 1999. On the Complexity of Policy Iteration. *In Proc. UAI 1999*, pp. 401–408, AUAI, 1999.

Daniel A. Spielman and Shang-Hua Teng, 2004. *Journal of the ACM*, 51(3):385–463, 2004.

John Fearnley, 2010. Exponential Lower Bounds for Policy Iteration. *In Proc. ICALP 2010*, pp. 551–562, Springer, 2010.

Thomas Dueholm Hansen and Uri Zwick, 2010. Lower bounds for Howard's algorithm for finding minimum mean-cost cycles. *In Proc. ISAAC 2010*, pp. 415–426, Springer 2010.

References

Omid Madani, Mikkel Thorup, and Uri Zwick, 2010. Discounted deterministic Markov decision processes and discounted all-pairs shortest paths. *ACM Transactions on Algorithms*, 6(2):33:1–33:25, 2010.

Thomas Dueholm Hansen, 2012. Worst-case Analysis of Strategy Iteration and the Simplex Method. PhD thesis, Department of Computer Science, Aarhus University, July 2012.

Romain Hollanders, Balázs Gerencsér, Jean-Charles Delvenne, 2012. The complexity of policy iteration is exponential for discounted Markov decision processes. *In Proc. CDC 2012*, pp. 5997–6002, IEEE, 2012.

Ian Post and Yinyu Ye. The Simplex Method is Strongly Polynomial for Deterministic Markov Decision Processes. *In Proc. SODA 2013*, pp.1465–1473, SIAM, 2013.

Romain Hollanders, Balázs Gerencsér, Jean-Charles Delvenne, and Raphaël M. Jungers, 2014. Improved bound on the worst case complexity of policy iteration. `http://arxiv.org/pdf/1410.7583v1.pdf`.

Balázs Gerencsér, Romain Hollanders, Jean-Charles Delvenne, and Raphaël M. Jungers, 2015. A complexity analysis of policy iteration through combinatorial matrices arising from unique sink orientations.`http://arxiv.org/pdf/1407.4293v2.pdf`.

Shivaram Kalyanakrishnan, Utkarsh Mall, and Ritish Goyal, 2016a. Batch-Switching Policy Iteration. *In Proc. IJCAI 2016*, pp. 3147–3153, AAAI Press, 2016.

Shivaram Kalyanakrishnan, Neeldhara Misra, and Aditya Gopalan, 2016b. Randomised Procedures for Initialising and Switching Actions in Policy Iteration. *In Proc. AAAI 2016*, pp. 3145–3151, AAAI Press, 2016.

Anchit Gupta and Shivaram Kalyanakrishnan, 2017. Improved Strong Worst-case Upper Bounds for MDP Planning. *In Proc. IJCAI 2017*, pp. 1788–1794, IJCAI, 2017.

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!