Solving Markov Decision Processes

Jefferson Huang

School of Operations Research and Information Engineering
Cornell University

February 8, 2018

Naval Postgraduate School

Monterey, CA

Searching for a Hidden Hostage

A hostage is known to be hidden somewhere in the city.

Searching for a Hidden Hostage

A hostage is known to be hidden somewhere in the city.

Searching for a Hidden Hostage

A hostage is known to be hidden somewhere in the city.

Objective: Find the hostage as quickly as possible.

Triaging & Treating Patients on a Battlefield

An explosion has left many people with severe injuries.

Triaging & Treating Patients on a Battlefield

An explosion has left many people with severe injuries.

- You are the only medical provider on the scene.
- For each patient, you can perform triage to roughly assess their condition, or perform treatment.

Triaging & Treating Patients on a Battlefield

An explosion has left many people with severe injuries.

- You are the only medical provider on the scene.
- For each patient, you can perform triage to roughly assess their condition, or perform treatment.

Objective: Save as many lives as possible.

Managing Blood Inventory

A blood bank serves several military hospitals.

Managing Blood Inventory

A blood bank serves several military hospitals.

- Random demand and donations.
- 8 blood types, some can substitute for others.
- Blood can only be stored for a few weeks.

Managing Blood Inventory

A blood bank serves several military hospitals.

- Random demand and donations.
- 8 blood types, some can substitute for others.
- Blood can only be stored for a few weeks.

Objective: Find the best way to use available blood to satisfy demand.

Common Features of the Problems

Common Features of the Problems

1. Decisions are made over time.

Common Features of the Problems

1. Decisions are made over time.

2. Decisions can be based on observations.

Common Features of the Problems

1. Decisions are made over time.

2. Decisions can be based on observations.

3. Depending on the observations, certain actions make more sense than others.

Common Features of the Problems

1. Decisions are made over time.
2. Decisions can be based on observations.
3. Depending on the observations, certain actions make more sense than others.
4. Each action has a cost.

Common Features of the Problems

1. Decisions are made over time.

2. Decisions can be based on observations.

3. Depending on the observations, certain actions make more sense than others.

4. Each action has a cost.

5. Taking an action will affect what you observe next.

Part 1

MDPs: Modeling Decision-Making

Modeling the Common Features

Modeling the Common Features

1. Decisions are made over time

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs

$$t = 0, 1, 2, 3, \ldots, T$$

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs

$$t = 0, 1, 2, 3, \ldots, T$$

2. Decisions can be based on observations

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs

$$t = 0, 1, 2, 3, \ldots, T$$

2. Decisions can be based on observations of the system state

$$x$$

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs
$$t = 0, 1, 2, 3, \ldots, T$$
2. Decisions can be based on observations of the system state
$$x$$
3. Depending on the observations, certain actions make more sense than others

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs

$$t = 0, 1, 2, 3, \ldots, T$$

2. Decisions can be based on observations of the system state

$$x$$

3. Depending on the observations, certain actions make more sense than others

$$a$$

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs
$$t = 0, 1, 2, 3, \ldots, T$$
2. Decisions can be based on observations of the system state
$$x$$
3. Depending on the observations, certain actions make more sense than others
$$a$$
4. Each action has a cost

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs
$$t = 0, 1, 2, 3, \ldots, T$$
2. Decisions can be based on observations of the system state
$$x$$
3. Depending on the observations, certain actions make more sense than others
$$a$$
4. Each action has a cost that depends on the current state:
$$c(x, a)$$

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs
$$t = 0, 1, 2, 3, \ldots, T$$
2. Decisions can be based on observations of the system state
$$x$$
3. Depending on the observations, certain actions make more sense than others
$$a$$
4. Each action has a cost that depends on the current state:
$$c(x, a)$$
5. Taking an action will affect what you observe next

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs
$$t = 0, 1, 2, 3, \ldots, T$$
2. Decisions can be based on observations of the system state
$$x$$
3. Depending on the observations, certain actions make more sense than others
$$a$$
4. Each action has a cost that depends on the current state:
$$c(x, a)$$
5. Taking an action will affect what you observe next

   via a probability distribution $\quad p(\cdot|x, a)$

Modeling the Common Features

1. Decisions are made over time, at discrete decision epochs

$$t = 0, 1, 2, 3, \ldots, T$$

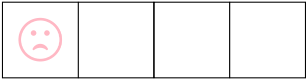
An **MDP** is defined by defining:

- the state set $\mathbb{X}$;
- the action set $A(x)$ for each state x;
- for each state x and action a,
  - the one-step cost $c(x, a)$, and
  - the transition probability distribution $p(\cdot|x, a)$ on the next state.

5. Taking an action will affect what you observe next

via a probability distribution $\quad p(\cdot|x, a)$

Searching for a Hidden Hostage: MDP



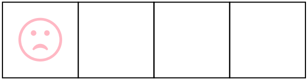
Time Elapsed: 0 hours

Only 1 location can be searched at a time.

1. P_i = posterior probability that the hostage is in location i
2. C_i = time needed to search location i
3. α_i = probability that search of location i gives a true positive
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (1/4 1/4 1/4 1/4)



Time Elapsed: 0 hours

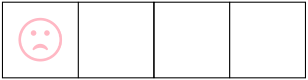
Action =

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (1/4 1/4 1/4 1/4)



Time Elapsed: 0 hours

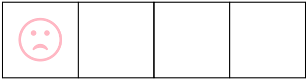
Action = $\uparrow$

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (3/10 1/10 3/10 3/10)



Time Elapsed: 1 hours

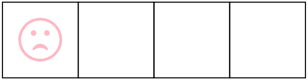
Action =

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (3/10 1/10 3/10 3/10)



Time Elapsed: 1 hours

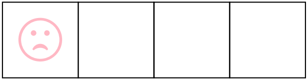
Action = ↑

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (1/8 1/8 3/8 3/8)



Time Elapsed: 2 hours

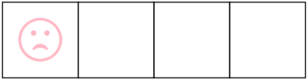
Action =

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State $= (\quad 1/8 \quad 1/8 \quad 3/8 \quad 3/8 \quad)$



Time Elapsed: 2 hours

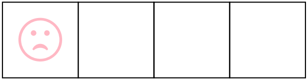
Action $= \qquad\qquad\qquad\qquad \uparrow$

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (1/6 1/6 1/2 1/6)



Time Elapsed: 4 hours

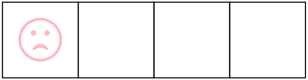
Action =

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (1/6 1/6 1/2 1/6)



Time Elapsed: 4 hours

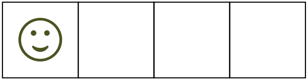
Action = ↑

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (1 0 0 0)



Time Elapsed: 5 hours

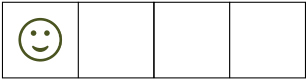
Action =

Only 1 location can be searched at a time.

1. Initially, $P_1 = P_2 = P_3 = P_4 = 1/4$.
2. Times to Search: $C_1 = 1,\ C_2 = 1,\ C_3 = 3,\ C_4 = 2$
3. True Positive Chances: $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 2/3$
4. No false positives.

Searching for a Hidden Hostage: MDP

State = (1 0 0 0)



Time Elapsed: 5 hours

Action =

MDP Formulation:

1. Decision Epoch = end of a search
2. State = $(P_1, P_2, P_3, P_4) = \mathbf{P}$
3. Action = location to search
4. $c(\mathbf{P}, i) = C_i$
5. $p(\cdot|\mathbf{P}, i) : \begin{cases} \text{find hostage} & \text{with probability } \alpha_i P_i \\ \text{update } \mathbf{P} \text{ with Bayes' Rule} & \text{with probability } 1 - \alpha_i P_i \end{cases}$

Control via a Policy

Q: Given the state, what action should be taken?

Control via a Policy

Q: Given the state, what action should be taken?

A: Act according to a **policy** φ.

Control via a Policy

Q: Given the state, what action should be taken?

A: Act according to a **policy** φ.

i.e., at each decision epoch t,

specify which action $\varphi_t(x)$ to take if state x is observed.

Control via a Policy

Q: Given the state, what action should be taken?

A: Act according to a **policy** φ.

i.e., at each decision epoch t,

specify which action $\varphi_t(x)$ to take if state x is observed.

e.g., in the hostage search problem,

search the location i maximizing $\frac{\alpha_i P_i}{C_i}$.

Optimal Policies

Q: Which policy should be followed?

Optimal Policies

Q: Which policy should be followed?

A: Follow an **optimal** policy.

Optimal Policies

Q: Which policy should be followed?

A: Follow an **optimal** policy.

Initial State: x

Optimal Policies

Q: Which policy should be followed?

A: Follow an **optimal** policy.

Initial State: x

Expected total cost of a policy φ:

$$v_T(x, \varphi) = \mathbb{E}\left[\sum_{t=0}^{T-1} c(x_t, \varphi_t(x_t))\right]$$

Optimal Policies

Q: Which policy should be followed?

A: Follow an **optimal** policy.

Initial State: x

Expected total cost of a policy φ:

$$v_T(x, \varphi) = \mathbb{E}\left[\sum_{t=0}^{T-1} c(x_t, \varphi_t(x_t))\right]$$

A policy is **optimal** if, for every initial state x, it

minimizes $v_T(x, \varphi)$ over all policies φ.

Computing an Optimal Policy

Assume:

- number of states is finite
- planning horizon T is finite

Computing an Optimal Policy

Assume:

- number of states is finite
- planning horizon T is finite (for now)

Computing an Optimal Policy

Assume:

- number of states is finite
- planning horizon T is finite (for now)

Q: How can an optimal policy be obtained?

Computing an Optimal Policy

Assume:

- number of states is finite
- planning horizon T is finite (for now)

Q: How can an optimal policy be obtained?

A: Value Iteration:

Computing an Optimal Policy

Assume:

- number of states is finite
- planning horizon T is finite (for now)

Q: How can an optimal policy be obtained?

A: Value Iteration:

1. Define $V_0(x) = 0$ for all states x.

Computing an Optimal Policy

Assume:

- number of states is finite
- planning horizon T is finite (for now)

Q: How can an optimal policy be obtained?

A: Value Iteration:

1. Define $V_0(x) = 0$ for all states x.
2. For $t = 1, 2, \ldots, T$,

$$\varphi^*_{T-t}(x) = \arg\min_{a \in A(x)} \left\{ c(x, a) + \sum_{y \in \mathbb{X}} p(y|x, a) V_{t-1}(y) \right\}$$

$$V_t(x) = \min_{a \in A(x)} \left[c(x, a) + \sum_{y \in \mathbb{X}} p(y|x, a) V_{t-1}(y) \right]$$

Computing an Optimal Policy

Assume:

- number of states is finite
- planning horizon T is finite (for now)

Q: How can an optimal policy be obtained?

Theorem:

The policy φ^* is optimal for the planning horizon T.

$$\varphi^*_{T-t}(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = \min_{a \in A(x)} \left[c(x,a) + \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right]$$

Infinite Horizon with Discounting

Q: What should the planning horizon be?

Infinite Horizon with Discounting

Q: What should the planning horizon be?

A: Assume that it is **infinite**.

Infinite Horizon with Discounting

Q: What should the planning horizon be?

A: Assume that it is **infinite**. (a simplification)

Infinite Horizon with Discounting

Q: What should the planning horizon be?

A: Assume that it is **infinite**. (a simplification)

Introduce a fixed **discount factor** β

Infinite Horizon with Discounting

Q: What should the planning horizon be?

A: Assume that it is **infinite**. (a simplification)

Introduce a fixed **discount factor** β (to ensure finiteness):

Infinite Horizon with Discounting

Q: What should the planning horizon be?

A: Assume that it is **infinite**. (a simplification)

Introduce a fixed **discount factor** β (to ensure finiteness):

$$v(x, \varphi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t c(x_t, \varphi_t(x_t))\right]$$

Infinite Horizon with Discounting

Q: What should the planning horizon be?

A: Assume that it is **infinite**. (a simplification)

Introduce a fixed **discount factor** β (to ensure finiteness):

$$v(x, \varphi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t c(x_t, \varphi_t(x_t))\right]$$

Possible Interpretations of β:

1. Reflects time-value of money.
2. After each decision, the problem terminates with probability $(1 - \beta)$.

Infinite Horizon with Discounting

Q: What should the planning horizon be?

A: Assume that it is **infinite**. (a simplification)

> **Theorem:**
>
> There exists an optimal policy that is **stationary** (doesn't depend on what epoch t it is).

Possible Interpretations of β:

1. Reflects time-value of money.
2. After each decision, the problem terminates with probability $(1 - \beta)$.

Computing an Optimal Policy: Infinite Horizon

Q: How can a stationary optimal policy be **computed**?

Computing an Optimal Policy: Infinite Horizon

Q: How can a stationary optimal policy be **computed**?

A: Value Iteration:

Computing an Optimal Policy: Infinite Horizon

Q: How can a stationary optimal policy be **computed**?

A: Value Iteration:

1. Define $V_0(x) = 0$ for all states x.

Computing an Optimal Policy: Infinite Horizon

Q: How can a stationary optimal policy be **computed**?

A: Value Iteration:

1. Define $V_0(x) = 0$ for all states x.
2. For $t = 1, 2, \ldots$,

$$\varphi_t^*(x) = \underset{a \in A(x)}{\arg\min} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = \min_{a \in A(x)} \left[c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right]$$

Computing an Optimal Policy: Infinite Horizon

Q: How can a stationary optimal policy be **computed**?

Theorem:

For some finite T, the stationary policy that always takes actions according to $\varphi_T^*(\cdot)$ is optimal.

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = \min_{a \in A(x)} \left[c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right]$$

Computing an Optimal Policy: Infinite Horizon

Q: How can a stationary optimal policy be **computed**?

Question:

Are there any guarantees on when value iteration will produce an optimal policy?

$$\varphi_t^*(x) = \operatorname*{arg\,min}_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = \min_{a \in A(x)} \left[c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right]$$

Part 2

Efficiency of Computing Optimal Policies

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word “abalone” in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

Which is better?

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

Which is better?

- "Linear Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned
$\uparrow$

Which is better?

- "Linear Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word “abalone” in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- “Linear Search”:

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search": 7 steps.

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word “abalone” in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

Which is better?

- “Linear Search”: 7 steps.
- “Binary Search”:

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search": 7 steps.
- "Binary Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search": 7 steps.
- "Binary Search":

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

$\uparrow$

Which is better?

- "Linear Search": 7 steps.
- "Binary Search": 3 steps.

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

Which is better?

- "Linear Search": 7 steps.
- "Binary Search": 3 steps.

What if we were searching for "aardvark"?

Assessing the Efficiency of Algorithms

Q: How can the efficiency of an algorithm be measured?

Example: Find the word "abalone" in a dictionary.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

Which is better?

- "Linear Search": 7 steps.
- "Binary Search": 3 steps.

What if we were searching for "aardvark"?

What if the computer is very fast? Very slow?

Assessing the Efficiency of Algorithms

A1: Look at the number of steps needed in the worst case.

Assessing the Efficiency of Algorithms

A1: Look at the number of steps needed in the worst case.

Example: Finding a word in a dictionary of length $N = 9$.

Assessing the Efficiency of Algorithms

A1: Look at the number of steps needed in the worst case.

Example: Finding a word in a dictionary of length $N = 9$.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

Assessing the Efficiency of Algorithms

A1: Look at the number of steps needed in the worst case.

Example: Finding a word in a dictionary of length $N = 9$.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

- Linear Search: at worst N steps.

Assessing the Efficiency of Algorithms

A1: Look at the number of steps needed in the worst case.

Example: Finding a word in a dictionary of length $N = 9$.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

- Linear Search: at worst N steps.
- Binary Search: at worst $\approx \log_2 N$ steps

Assessing the Efficiency of Algorithms

A1: Look at the number of steps needed in the worst case.

Example: Finding a word in a dictionary of length $N = 9$.

aardvark aardwolf aaron aback abacus abaft abalone abandon abandoned

- Linear Search: at worst N steps.
- Binary Search: at worst $\approx \log_2 N$ steps

If N is large, Linear Search may need many more steps.

Assessing the Efficiency of Algorithms

A2: Look at the **growth rate** of the number of steps needed in the worst case.

Assessing the Efficiency of Algorithms

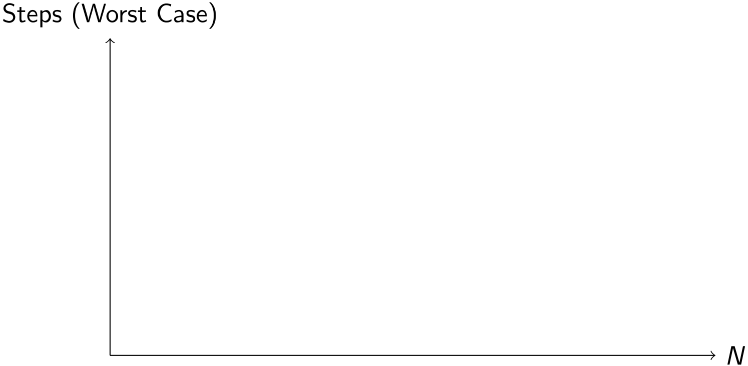
A2: Look at the **growth rate** of the number of steps needed in the worst case.

Example (Cont.): As the dictionary size N grows, the worst case number of steps of Linear Search grows much faster than that of Binary Search.

Assessing the Efficiency of Algorithms

A2: Look at the **growth rate** of the number of steps needed in the worst case.

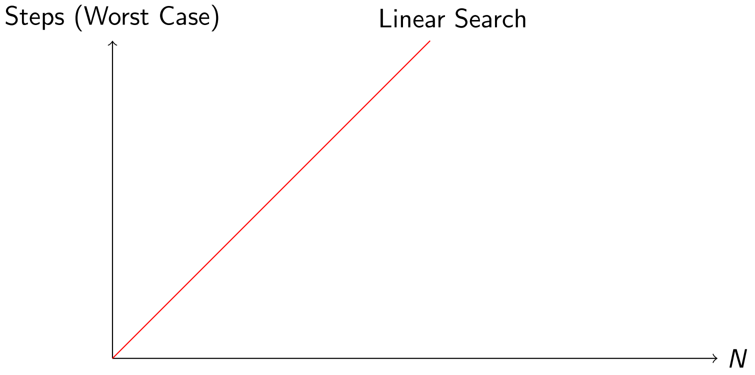
Example (Cont.): As the dictionary size N grows, the worst case number of steps of Linear Search grows much faster than that of Binary Search.



Assessing the Efficiency of Algorithms

A2: Look at the **growth rate** of the number of steps needed in the worst case.

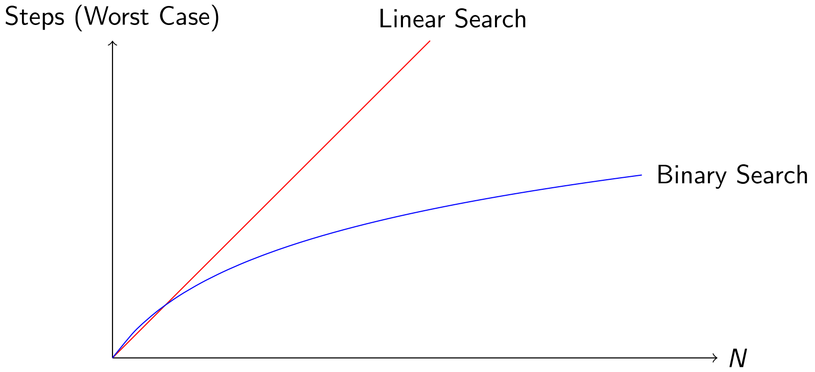
Example (Cont.): As the dictionary size N grows, the worst case number of steps of Linear Search grows much faster than that of Binary Search.



Assessing the Efficiency of Algorithms

A2: Look at the **growth rate** of the number of steps needed in the worst case.

Example (Cont.): As the dictionary size N grows, the worst case number of steps of Linear Search grows much faster than that of Binary Search.



Complexity of Algorithms for MDPs

- m = number of states in the MDP

Complexity of Algorithms for MDPs

- m = number of states in the MDP
- L = number of bits needed to encode the MDP data

Complexity of Algorithms for MDPs

- $m =$ number of states in the MDP
- $L =$ number of bits needed to encode the MDP data
- $S(m, L) =$ number of arithmetic operations that a given algorithm needs to return an optimal policy for the MDP

Complexity of Algorithms for MDPs

- m = number of states in the MDP
- L = number of bits needed to encode the MDP data
- $S(m, L)$ = number of arithmetic operations that a given algorithm needs to return an optimal policy for the MDP

Definition: An algorithm for computing an optimal policy is

Complexity of Algorithms for MDPs

- m = number of states in the MDP
- L = number of bits needed to encode the MDP data
- $S(m, L)$ = number of arithmetic operations that a given algorithm needs to return an optimal policy for the MDP

Definition: An algorithm for computing an optimal policy is

1. **weakly polynomial** if $S(m, L)$ can be bounded above by a polynomial function of m and L;

Complexity of Algorithms for MDPs

- m = number of states in the MDP
- L = number of bits needed to encode the MDP data
- $S(m, L)$ = number of arithmetic operations that a given algorithm needs to return an optimal policy for the MDP

Definition: An algorithm for computing an optimal policy is

1. **weakly polynomial** if $S(m, L)$ can be bounded above by a polynomial function of m and L;
2. **strongly polynomial** if $S(m, L)$ can be bounded above by a polynomial function of m only.

Complexity of Algorithms for MDPs

- m = number of states in the MDP
- L = number of bits needed to encode the MDP data

What happens as $L \to \infty$? (e.g., costs get large)

1. weakly polynomial $\implies$ bound goes to ∞
2. strongly polynomial. $\implies$ bound is *unchanged*

1. **weakly polynomial** if $S(m, L)$ can be bounded above by a polynomial function of m and L;
2. **strongly polynomial** if $S(m, L)$ can be bounded above by a polynomial function of m only.

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Q: Is value iteration weakly polynomial?

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Q: Is value iteration weakly polynomial?

Each iteration:

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\} \quad \sim m^3 \text{ steps}$$

$$V_t(x) = \min_{a \in A(x)} \left[c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right] \quad \sim m^3 \text{ steps}$$

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Theorem: (Tseng, 1990)
Value iteration needs at most[a]

$$\mathcal{O}\left(m[\log(m) + L]\right)$$

iterations to return an optimal policy.

[a] $f(x) = \mathcal{O}(g(x))$ if $\limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$

$$V_t(x) = \min_{a\in A(x)} \left[c(x,a) + \beta \sum_{y\in\mathbb{X}} p(y|x,a) V_{t-1}(y)\right] \quad \sim m^3 \text{ steps}$$

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Theorem: (Tseng, 1990)
Value iteration needs at most[a]

$$\mathcal{O}\left(m[\log(m) + L]\right)$$

iterations to return an optimal policy.

[a] $f(x) = \mathcal{O}(g(x))$ if $\limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty$

Conclusion:
Value iteration is weakly polynomial.

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Q: Is value iteration strongly polynomial?

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Q: Is value iteration strongly polynomial?

Each iteration needs $\mathcal{O}(m^3)$ steps.

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Q: Is value iteration strongly polynomial?

Each iteration needs $\mathcal{O}(m^3)$ steps.

To answer **Q**, it suffices to answer **Q'**:

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Q: Is value iteration strongly polynomial?

Each iteration needs $\mathcal{O}(m^3)$ steps.

To answer **Q**, it suffices to answer **Q'**:

Q': Can the number of iterations needed to return an optimal policy be bounded by a polynomial in m only?

Efficiency of Value Iteration

Assume the discount factor β is a constant.

Q: Is value iteration strongly polynomial?

Theorem: (H. et al, 2014)

Value iteration is not strongly polynomial.

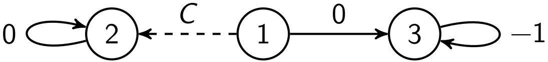
Q': Can the number of iterations needed to return an optimal policy be bounded by a polynomial in m only?

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.

Value Iteration:

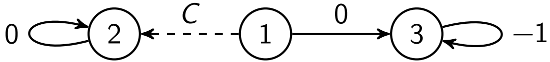
1. Define $V_0(x) = 0$ for all states x.
2. For $t = 1, 2, \ldots,$

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = \min_{a \in A(x)} \left[c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right]$$

Proof that Value Iteration is Not Strongly Polynomial

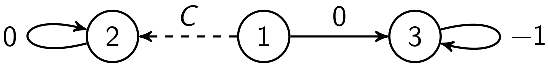
Discount Factor: $\beta < 1$.



Iteration 1:

Proof that Value Iteration is Not Strongly Polynomial

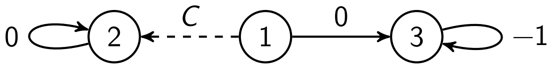
Discount Factor: $\beta < 1$.



Iteration 1: $V_1(1) = \min\{C, 0\}$

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.

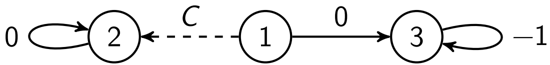


Iteration 1: $V_1(1) = \min\{C, 0\}$

Iteration 2:

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.

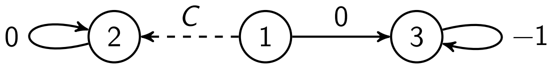


Iteration 1: $V_1(1) = \min\{C, 0\}$

Iteration 2: $V_2(1) = \min\{C, \beta V_1(3)\}$

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



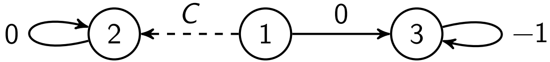
Iteration 1: $V_1(1) = \min\{C, 0\}$

Iteration 2: $V_2(1) = \min\{C, \beta V_1(3)\}$

Iteration 3:

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



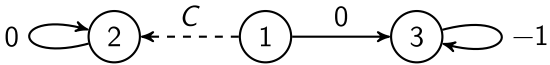
Iteration 1: $V_1(1) = \min\{C, 0\}$

Iteration 2: $V_2(1) = \min\{C, \beta V_1(3)\}$

Iteration 3: $V_3(1) = \min\{C, \beta V_2(3)\}$

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



Iteration 1: $V_1(1) = \min\{C, 0\}$

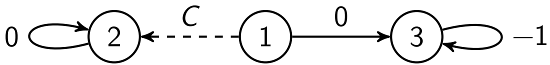
Iteration 2: $V_2(1) = \min\{C, \beta V_1(3)\}$

Iteration 3: $V_3(1) = \min\{C, \beta V_2(3)\}$

$\vdots$ $\vdots$

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



Iteration 1: $V_1(1) = \min\{C, 0\}$

Iteration 2: $V_2(1) = \min\{C, \beta V_1(3)\}$

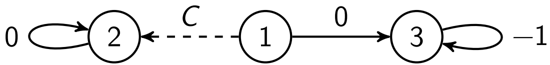
Iteration 3: $V_3(1) = \min\{C, \beta V_2(3)\}$

$\vdots$ $\vdots$

Iteration t:

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



Iteration 1: $V_1(1) = \min\{C, 0\}$

Iteration 2: $V_2(1) = \min\{C, \beta V_1(3)\}$

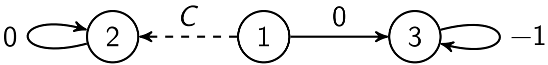
Iteration 3: $V_3(1) = \min\{C, \beta V_2(3)\}$

$\vdots$ $\vdots$

Iteration t: $V_t(1) = \min\{C, \beta V_{t-1}(3)\}$

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



Observations:

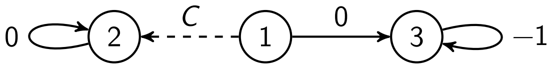
1. $V_t(3) = -(1 + \beta + \beta^2 + \cdots + \beta^{t-1}) = -\frac{1-\beta^t}{1-\beta}$
2. Once "right" is selected, "left" will never again be selected.

$\vdots \qquad\qquad \vdots$

Iteration t: $\qquad V_t(1) = \min\{C, \beta V_{t-1}(3)\}$

Proof that Value Iteration is Not Strongly Polynomial

Discount Factor: $\beta < 1$.



Idea: Given $N \geqslant 1$, select C so that:

1. "right" is optimal, and "left" is not.
2. On iterations $t = 1, \ldots, N$, "left" is selected.

$\vdots$ $\qquad\qquad$ $\vdots$

Iteration t: $\qquad V_t(1) = \min\{C, \beta V_{t-1}(3)\}$

Proof that Value Iteration is Not Strongly Polynomial

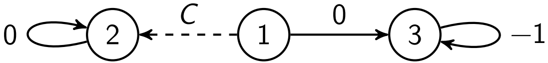
Given $N \geqslant 1$,

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.

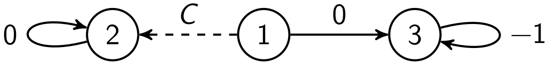
Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Proof that Value Iteration is Not Strongly Polynomial

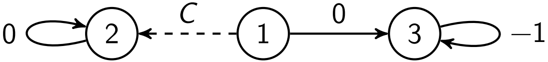
Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Iteration 1:

Proof that Value Iteration is Not Strongly Polynomial

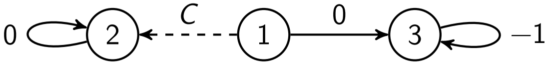
Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Iteration 1: $V_1(1) = \min\{C, 0\}$, $\varphi_1^*(1) = \text{"left"}$

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.

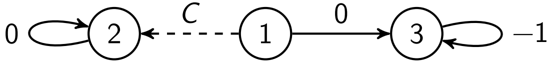


Iteration 1: $V_1(1) = \min\{C, 0\}$, $\varphi_1^*(1) =$ "left"

Iteration 2:

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.

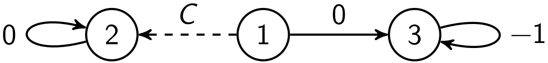


Iteration 1: $V_1(1) = \min\{C, 0\}$, $\varphi_1^*(1) =$ "left"

Iteration 2: $V_2(1) = \min\{C, -\beta\}$, $\varphi_2^*(1) =$ "left"

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



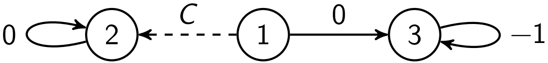
Iteration 1: $V_1(1) = \min\{C, 0\}$, $\varphi_1^*(1) = \text{"left"}$

Iteration 2: $V_2(1) = \min\{C, -\beta\}$, $\varphi_2^*(1) = \text{"left"}$

$\vdots$ $\vdots$

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Iteration 1: $V_1(1) = \min\{C, 0\}$, $\varphi_1^*(1) =$ "left"

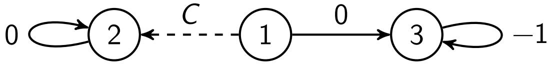
Iteration 2: $V_2(1) = \min\{C, -\beta\}$, $\varphi_2^*(1) =$ "left"

$\vdots$ $\vdots$

Iteration N:

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Iteration 1: $V_1(1) = \min\{C, 0\}$, $\varphi_1^*(1) =$ "left"

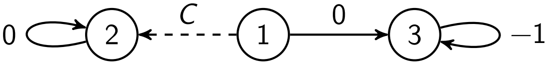
Iteration 2: $V_2(1) = \min\{C, -\beta\}$, $\varphi_2^*(1) =$ "left"

$\vdots$ $\vdots$

Iteration N: $V_N(1) = \min\left\{C, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right\}$, $\varphi_N^*(1) =$ "left"

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Iteration 1: $V_1(1) = \min\{C, 0\}, \quad \varphi_1^*(1) = \text{“left”}$

Iteration 2: $V_2(1) = \min\{C, -\beta\}, \quad \varphi_2^*(1) = \text{“left”}$

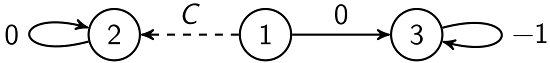
$\vdots \qquad\qquad \vdots$

Iteration N: $V_N(1) = \min\left\{C, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right\}, \quad \varphi_N^*(1) = \text{“left”}$

Iteration $N+1$:

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Iteration 1: $V_1(1) = \min\{C, 0\}$, $\varphi_1^*(1) =$ "left"

Iteration 2: $V_2(1) = \min\{C, -\beta\}$, $\varphi_2^*(1) =$ "left"

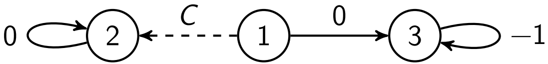
$\vdots$ $\vdots$

Iteration N: $V_N(1) = \min\left\{C, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right\}$, $\varphi_N^*(1) =$ "left"

Iteration $N+1$: $V_{N+1}(1) = \min\left\{C, -\frac{\beta(1-\beta^{N})}{1-\beta}\right\}$, $\varphi_{N+1}^*(1) =$ "right"

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



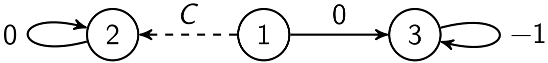
> **Theorem:** (H., 2016)
>
> For every $N \geqslant 1$, there is an MDP with $m = 4$ state-action pairs for which value iteration needs at least N iterations to return the optimal policy.

Iteration N: $V_N(1) = \min\left\{C, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right\}$, $\varphi^*_N(1) = \text{“left”}$

Iteration $N+1$: $V_{N+1}(1) = \min\left\{C, -\frac{\beta(1-\beta^{N})}{1-\beta}\right\}$, $\varphi^*_{N+1}(1) = \text{“right”}$

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Conclusion:

Value iteration is not strongly polynomial.

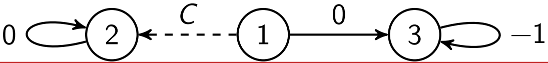
$\vdots$ $\qquad\qquad\qquad\qquad$ $\vdots$

Iteration N: $\quad V_N(1) = \min\left\{C, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right\}, \quad \varphi_N^*(1) = \text{“left”}$

Iteration $N+1$: $\quad V_{N+1}(1) = \min\left\{C, -\frac{\beta(1-\beta^{N})}{1-\beta}\right\}, \quad \varphi_{N+1}^*(1) = \text{“right”}$

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.



Conclusion:

Value iteration is not strongly polynomial.

(If it were, then value iteration should return the optimal policy for the example in at most a constant number of iterations.)

Iteration N: $V_N(1) = \min\left\{C, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right\}$, $\varphi^*_N(1) =$ "left"

Iteration $N+1$: $V_{N+1}(1) = \min\left\{C, -\frac{\beta(1-\beta^{N})}{1-\beta}\right\}$, $\varphi^*_{N+1}(1) =$ "right"

Proof that Value Iteration is Not Strongly Polynomial

Given $N \geqslant 1$, select any C on the interval $\left(-\frac{\beta}{1-\beta}, -\frac{\beta(1-\beta^{N-1})}{1-\beta}\right)$.

Stronger Conclusion:

The number of iterations needed by value iteration to return an optimal policy cannot be bounded above by any function of m only.

(If it cound, then value iteration should return the optimal policy for the example in at most a constant number of iterations.)

t”

Iteration $N+1$: $V_{N+1}(1) = \min\left\{C, -\frac{\beta(1-\beta^{N})}{1-\beta}\right\}$, $\varphi^*_{N+1}(1) =$ “right”

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

A: Yes!

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

A: Yes! **Policy Iteration:**

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

A: Yes! **Policy Iteration:** First, select any policy φ_0^*.

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

A: Yes! **Policy Iteration:** First, select any policy φ_0^*.

1. Define $V_0(x) = v(x, \varphi_0^*)$ for all states x.

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

A: Yes! **Policy Iteration:** First, select any policy φ_0^*.

1. Define $V_0(x) = v(x, \varphi_0^*)$ for all states x.
2. For $t = 1, 2, \ldots$,

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = v(x, \varphi_t^*)$$

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

> **Theorem:** (Ye, 2011)
>
> The policy iteration algorithm is strongly polynomial.

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = v(x, \varphi_t^*)$$

Strongly Polynomial Algorithms for MDPs

Assume the discount factor β is a constant.

Q: Is there a strongly polynomial algorithm for MDPs?

> **Theorem:** (Ye, 2011)
>
> The policy iteration algorithm is strongly polynomial.

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = v(x, \varphi_t^*) \quad \leftarrow \text{hard if the state set is large}$$

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap, but may require more iterations.

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap, but may require more iterations.

Policy Iteration: Each iteration is expensive

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap, but may require more iterations.

Policy Iteration: Each iteration is expensive, but may require fewer iterations.

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap, but may require more iterations.

Policy Iteration: Each iteration is expensive, but may require fewer iterations.

Q: Is there a way to combine the good qualities of both?

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap, but may require more iterations.

Policy Iteration: Each iteration is expensive, but may require fewer iterations.

Q: Is there a way to combine the good qualities of both?

A: One approach is **Modified Policy Iteration**: First, select $M \geqslant 0$.

1. Define $V_0(x) = 0$ for all states x.
2. For $t = 1, 2, \ldots$,

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = T_{\varphi_t^*}^M V_{t-1}(x)$$

Value Iteration vs. Policy Iteration

$T^{M}_{\varphi^*_t}$ **"interpolates"** between VI and PI:

Value Iteration:

$$T^{0}_{\varphi^*_t} V_{t-1}(x) = \min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

Policy Iteration:

$$\lim_{M \to \infty} T^{M}_{\varphi^*_t} V_{t-1}(x) = v(x, \varphi^*_t)$$

In Between: $M = 1, 2, \ldots$

$$T^{M}_{\varphi^*_t} V_{t-1}(x) = \mathbb{E}\left[\sum_{n=0}^{M} \beta^n c(x_n, \varphi^*_t(x_n)) + \beta^{M+1} V_{t-1}(x_{T+1}) \right]$$

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap, but may require more iterations.

Policy Iteration: Each iteration is expensive, but may require fewer iterations.

> **Question:**
>
> Is modified policy iteration strongly polynomial, for some M?

2. For $t = 1, 2, \ldots,$

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x, a) + \beta \sum_{y \in \mathbb{X}} p(y|x, a) V_{t-1}(y) \right\}$$

$$V_t(x) = T_{\varphi_t^*}^M V_{t-1}(x)$$

Value Iteration vs. Policy Iteration

Value Iteration: Each iteration is cheap, but may require more iterations.

Policy Iteration: Each iteration is expensive, but may require fewer iterations.

Theorem: (H. et al, 2014)

Modified policy iteration is not strongly polynomial for any M.

2. For $t = 1, 2, \ldots,$

$$\varphi_t^*(x) = \arg\min_{a \in A(x)} \left\{ c(x,a) + \beta \sum_{y \in \mathbb{X}} p(y|x,a) V_{t-1}(y) \right\}$$

$$V_t(x) = T_{\varphi_t^*}^M V_{t-1}(x)$$

Theoretical Efficiency of Algorithms for MDPs

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.
(via linear programming)

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.
(via linear programming)

Policy iteration shown to be weakly polynomial in **1986**.

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.
(via linear programming)

Policy iteration shown to be weakly polynomial in **1986**.

Value iteration (and hence modified policy iteration) shown to be weakly polynomial in **1990**.

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.
(via linear programming)

Policy iteration shown to be weakly polynomial in **1986**.

Value iteration (and hence modified policy iteration) shown to be weakly polynomial in **1990**.

Policy iteration shown to be strongly polynomial in **2011**.

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.
(via linear programming)

Policy iteration shown to be weakly polynomial in **1986**.

Value iteration (and hence modified policy iteration) shown to be weakly polynomial in **1990**.

Policy iteration shown to be strongly polynomial in **2011**.

Value iteration and modified policy iteration shown to be not strongly polynomial in **2014**.

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.
(via linear programming)

Summary:

We showed that there is a **stark difference** between value iteration (and modified policy iteration) and policy iteration.

Policy iteration shown to be strongly polynomial in **2011**.

Value iteration and modified policy iteration shown to be not strongly polynomial in **2014**.

Theoretical Efficiency of Algorithms for MDPs

Known to be solvable in weakly polynomial time since the **1980s**.
(via linear programming)

Question:

Given an MDP, which algorithm should be used?

Policy iteration shown to be strongly polynomial in **2011**.

Value iteration and modified policy iteration shown to be not strongly polynomial in **2014**.

Part 3

Computing Optimal Policies in Practice

Solving MDPs in Practice

Policy iteration should be used if possible.

Solving MDPs in Practice

Policy iteration should be used if possible.

- clear stopping criterion (unlike value iteration)

Solving MDPs in Practice

Policy iteration should be used if possible.

- clear stopping criterion (unlike value iteration)
- typically converges quickly

Solving MDPs in Practice

Policy iteration should be used if possible.

- clear stopping criterion (unlike value iteration)
- typically converges quickly

(e.g., $\sim 10,000$ states, ~ 100 actions per state)

Solving MDPs in Practice

Policy iteration should be used if possible.

- clear stopping criterion (unlike value iteration)
- typically converges quickly

(e.g., $\sim 10,000$ states, ~ 100 actions per state)

If policy iteration is infeasible, use **modified policy iteration**.

Solving MDPs in Practice

Policy iteration should be used if possible.

- clear stopping criterion (unlike value iteration)
- typically converges quickly

(e.g., $\sim 10,000$ states, ~ 100 actions per state)

If policy iteration is infeasible, use **modified policy iteration**.

- converges faster than value iteration

Solving MDPs in Practice

Policy iteration should be used if possible.

- clear stopping criterion (unlike value iteration)
- typically converges quickly

(e.g., $\sim 10,000$ states, ~ 100 actions per state)

If policy iteration is infeasible, use **modified policy iteration**.

- converges faster than value iteration
- not much more computationally expensive than value iteration

Solving MDPs in Practice

Policy iteration should be used if possible.

- clear stopping criterion (unlike value iteration)
- typically converges quickly

(e.g., $\sim 10,000$ states, ~ 100 actions per state)

If policy iteration is infeasible, use **modified policy iteration**.

- converges faster than value iteration
- not much more computationally expensive than value iteration

Otherwise, **approximate** methods are needed.

- approximate versions of value iteration, policy iteration

Example: Controlled Queue

At most N customers in the queue.

Example: Controlled Queue

At most N customers in the queue.

Finite number of possible service rates q

Example: Controlled Queue

At most N customers in the queue.

Finite number of possible service rates q

In each decision epoch:

Example: Controlled Queue

At most N customers in the queue.

Finite number of possible service rates q

In each decision epoch:

1. an arrival arrives with probability p;

Example: Controlled Queue

At most N customers in the queue.

Finite number of possible service rates q

In each decision epoch:

1. an arrival arrives with probability p;
2. if there is a customer, a service completion occurs with probability q.

Example: Controlled Queue

At most N customers in the queue.

Finite number of possible service rates q

In each decision epoch:

1. an arrival arrives with probability p;
2. if there is a customer, a service completion occurs with probability q.

Cost incurred if there are x customers and service rate q is used:

$$g(x, q) = x + 60q^3$$

Example: Controlled Queue

At most N customers in the queue.

Finite number of possible service rates q

In each decision epoch:

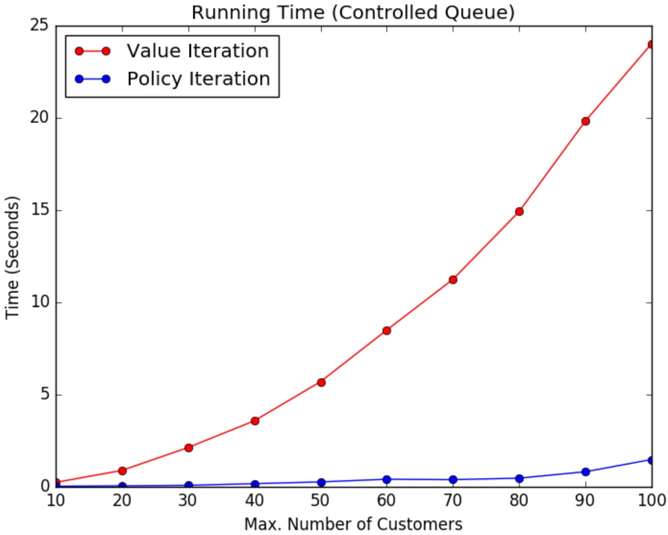
1. an arrival arrives with probability p;
2. if there is a customer, a service completion occurs with probability q.

Cost incurred if there are x customers and service rate q is used:

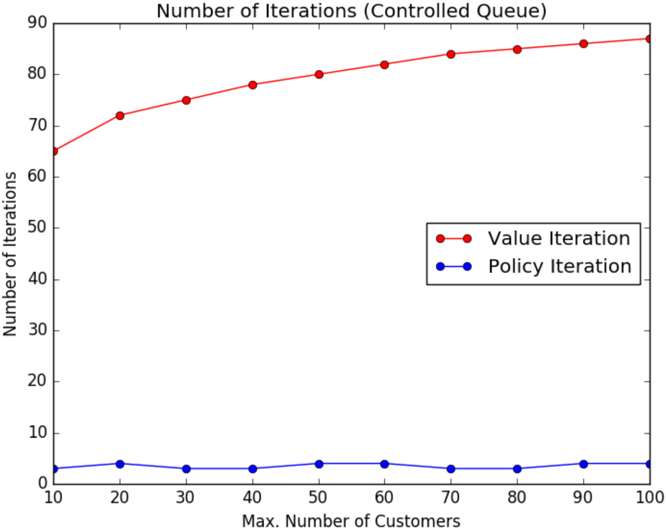
$$g(x, q) = x + 60q^3$$

Objective: Control the service rate to minimize the expected discounted total cost. (discount factor $= 0.9$)

Running Time: Value Iteration vs. Policy Iteration



Number of Iterations: Value Iteration vs. Policy Iteration



Part 4

Future Research

Balancing Basic and Applied Research

Balancing Basic and Applied Research

Basic:

Balancing Basic and Applied Research

Basic:

1. Existence of optimal policies for MDPs with general state and action sets (H. et al, 2017 & 2018)

Balancing Basic and Applied Research

Basic:

1. Existence of optimal policies for MDPs with general state and action sets (H. et al, 2017 & 2018)
2. Computational complexity of average-cost MDPs (H. et al, 2013 & 2017)

Balancing Basic and Applied Research

Basic:

1. Existence of optimal policies for MDPs with general state and action sets (H. et al, 2017 & 2018)
2. Computational complexity of average-cost MDPs (H. et al, 2013 & 2017)

Applied:

Balancing Basic and Applied Research

Basic:

1. Existence of optimal policies for MDPs with general state and action sets (H. et al, 2017 & 2018)
2. Computational complexity of average-cost MDPs (H. et al, 2013 & 2017)

Applied:

1. Joint maintenance and scheduling (e.g., in semiconductor manufacturing) (H. et al, 2018)

Balancing Basic and Applied Research

Basic:

1. Existence of optimal policies for MDPs with general state and action sets (H. et al, 2017 & 2018)
2. Computational complexity of average-cost MDPs (H. et al, 2013 & 2017)

Applied:

1. Joint maintenance and scheduling (e.g., in semiconductor manufacturing) (H. et al, 2018)
2. Inventory management (H. et al., 2018)

Balancing Basic and Applied Research

Basic:

1. Existence of optimal policies for MDPs with general state and action sets (H. et al, 2017 & 2018)
2. Computational complexity of average-cost MDPs (H. et al, 2013 & 2017)

Applied:

1. Joint maintenance and scheduling (e.g., in semiconductor manufacturing) (H. et al, 2018)
2. Inventory management (H. et al., 2018)
3. **Sequential decision-making in military operations**

Thank You!