

# Planning and Optimization

## F3. Policy Iteration

Malte Helmert and Thomas Keller

Universität Basel

December 02, 2019

# Planning and Optimization

December 02, 2019 — F3. Policy Iteration

F3.1 Introduction

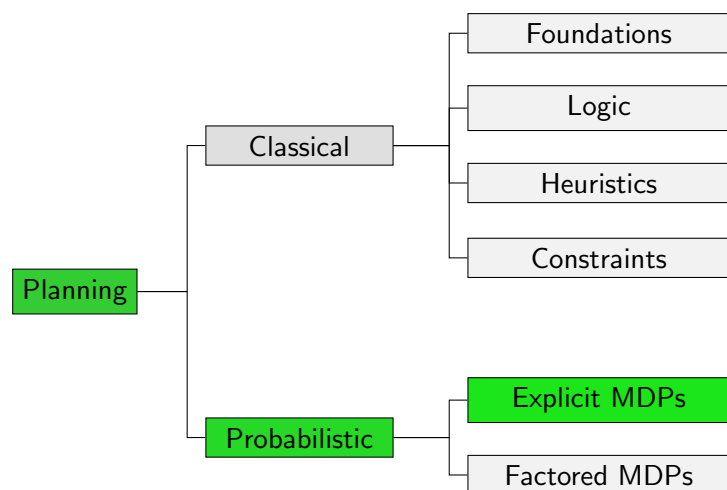
F3.2 Policy Evaluation

F3.3 Policy Improvement

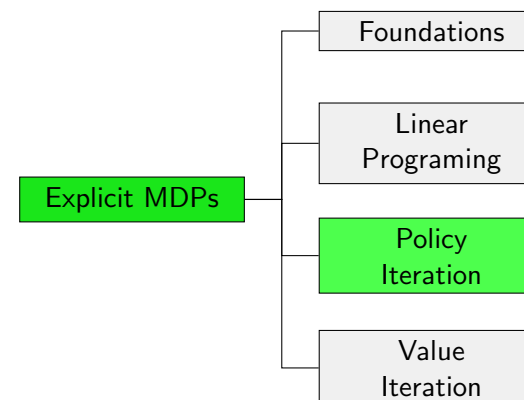
F3.4 Policy Iteration

F3.5 Summary

## Content of this Course



## Content of this Course: Explicit MDPs



## F3.1 Introduction

## Limitations of LPs in Practice

LP computes **optimal policy** in time **polynomial** in  $|S| \cdot |L|$

Possible issues in practice:

- ▶ LPs often **too expensive** even for small MDPs
- ▶ LP solver usage **prohibited**
- ▶ **More expressive model** required (e.g. continuous state space)

**Policy Iteration (PI)** is suitable alternative. PI has 2 components:

- ▶ **Policy Evaluation**
- ▶ **Policy Improvement**

## F3.2 Policy Evaluation

## Reminder: Value Functions for SSPs

### Definition (Value Functions for SSPs)

Let  $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$  be an SSP and  $\pi$  be an executable policy for  $\mathcal{T}$ .

The **state-value**  $V_\pi(s)$  of  $s$  under  $\pi$  is defined as

$$V_\pi(s) := \begin{cases} 0 & \text{if } s \in S_\star \\ Q_\pi(s, \pi(s)) & \text{otherwise,} \end{cases}$$

where the **action-value**  $Q_\pi(s, \ell)$  of  $s$  and  $\ell$  under  $\pi$  is defined as

$$Q_\pi(s, \ell) := c(\ell) + \sum_{s' \in \text{succ}(s, \ell)} T(s, \ell, s') \cdot V_\pi(s').$$

The state-value  $V_\pi(s)$  describes the **expected cost** of applying  $\pi$  in SSP  $\mathcal{T}$ , starting from  $s$ .

## Policy Evaluation: Implementations

Computing  $V_\pi$  for a given policy  $\pi$  is called **policy evaluation**.

There are several algorithms for policy evaluation:

- 1 **Linear Program**

## Reminder: LP for Expected Cost in SSP

### Variables

Non-negative variable  $\text{ExpCost}_s$  for each state  $s$

### Objective

Maximize  $\text{ExpCost}_{s_0}$

### Subject to

$$\text{ExpCost}_{s_*} = 0 \quad \text{for all goal states } s_*$$

$$\text{ExpCost}_s \leq \left( \sum_{s' \in S} T(s, \ell, s') \cdot \text{ExpCost}_{s'} \right) + c(\ell)$$

for all  $s \in S$  and  $\ell \in L(s)$

## LP for Policy Evaluation in SSP

### Variables

Non-negative variable  $\text{ExpCost}_s$  for each state  $s$

### Objective

Maximize  $\text{ExpCost}_{s_0}$

### Subject to

$$\text{ExpCost}_{s_*} = 0 \quad \text{for all goal states } s_*$$

$$\text{ExpCost}_s \leq \left( \sum_{s' \in S} T(s, \pi(s), s') \cdot \text{ExpCost}_{s'} \right) + c(\pi(s))$$

for all  $s \in S$  and  $\ell \in L(s)$

## Policy Evaluation via LP

- ▶ is polynomial in  $|S|$
- ▶ difference between polynomial in  $|S|$  and polynomial in  $|S| \cdot |L|$  is sometimes **relevant in practice**
- ▶ but often this is not the case
- ▶ other practical limitations also apply here

↔ Require policy evaluation without LP

## Policy Evaluation: Implementations

Computing  $V_\pi$  for a given policy  $\pi$  is called **policy evaluation**.

There are several algorithms for policy evaluation:

- 1 Linear Program
- 2 Backward Induction

## Example: Backward Induction in Deterministic SSP

5	$\Rightarrow$ 5.00	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_*$ 0.00
4	$\Rightarrow$ 6.00	$\uparrow$ 3.00	$\uparrow$ 4.00	$\uparrow$ 3.00
3	$\Rightarrow$ 7.00	$\uparrow$ 4.00	$\Leftarrow$ 5.00	$\Leftarrow$ 8.00
2	$\uparrow$ 10.00	$\uparrow$ 7.00	$\uparrow$ 6.00	$\Leftarrow$ 9.00
1	$\Rightarrow^{s_0}$ 9.00	$\Rightarrow$ 8.00	$\uparrow$ 7.00	$\Leftarrow$ 10.00
	1	2	3	4

- cost of 3 to move from striped cells (cost is 1 otherwise)

## Policy Evaluation via Backward Induction

- is linear in  $|S|$
- but restricted to special cases

↔ When is policy evaluation via backward induction possible?

In deterministic planning problems?

## Example: Backward Induction in Probabilistic SSP

5	$\Rightarrow$ 5.00	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_*$ 0.00
4	$\Rightarrow$ 6.00	$\uparrow$ 3.00	$\uparrow$ 2.80	$\uparrow$ 3.00
3	$\Rightarrow$ 7.00	$\uparrow$ 4.00	$\Leftarrow$ 5.00	$\Leftarrow$ 8.00
2	$\uparrow$ 10.00	$\uparrow$ 7.00	$\uparrow$ 6.00	$\Leftarrow$ 9.00
1	$\Rightarrow^{s_0}$ 9.00	$\Rightarrow$ 8.00	$\uparrow$ 7.00	$\Leftarrow$ 10.00
	1	2	3	4

- cost of 3 to move from striped cells (cost is 1 otherwise)
- probability of 0.4 to " $\Rightarrow$ " in gray cell

## Policy Evaluation via Backward Induction

- ▶ is linear in  $|S|$
- ▶ but restricted to special cases

↔ When is policy evaluation via backward induction possible?

In deterministic planning problems?

No, policy must be **acyclic**.

## Policy Evaluation: Implementations

Computing  $V_\pi$  for a given policy  $\pi$  is called **policy evaluation**.

There are several algorithms for policy evaluation:

- 1 Linear Program
- 2 Backward Induction for acyclic policies

## Backward Induction: Algorithm

Backward Induction for SSP  $\mathcal{T}$  and complete policy  $\pi$

initialize  $V_\pi(s) := \text{none}$  for all  $s \in S$

**while** there is a  $s \in S$  with  $V_\pi(s) = \text{none}$ :

  pick  $s \in S$  with  $V_\pi(s) = \text{none}$  and

$V_\pi(s') \neq \text{none}$  for all  $s' \in \text{succ}(s, \pi(s))$

  set  $V_\pi(s) := c(\pi(s)) + \sum_{s' \in S} T(s, \pi(s), s') \cdot V_\pi(s')$

**return**  $V_\pi$

## Policy Evaluation: Implementations

Computing  $V_\pi$  for a given policy  $\pi$  is called **policy evaluation**.

There are several algorithms for policy evaluation:

- 1 Linear Program
- 2 Backward Induction for acyclic policies
- 3 Iterative Policy Evaluation

## Iterative Policy Evaluation: Idea

- ▶ impossible to compute state-values in one sweep over the state space in presence of cycles
- ▶ start with arbitrary state-value function  $\hat{V}_\pi^0$
- ▶ treat state-value function as update rule

$$\hat{V}_\pi^i(s) = c(\pi(s)) + \sum_{s' \in S} T(s, \pi(s), s') \cdot \hat{V}_\pi^{i-1}(s')$$

- ▶ apply update rule iteratively
- ▶ until state-values have converged

## Iterative Policy Evaluation for SSPs: Example

5	$\Rightarrow$ 0.00	$\Rightarrow$ 0.00	$\Rightarrow$ 0.00	$s_x$ 0.00	$\hat{V}_\pi^0$
4	$\Rightarrow$ 0.00	$\uparrow$ 0.00	$\uparrow$ 0.00	$\uparrow$ 0.00	
3	$\Rightarrow$ 0.00	$\uparrow$ 0.00	$\Leftarrow$ 0.00	$\Leftarrow$ 0.00	
2	$\uparrow$ 0.00	$\uparrow$ 0.00	$\uparrow$ 0.00	$\Leftarrow$ 0.00	
1	$\Rightarrow^{s_0}$ 0.00	$\Rightarrow$ 0.00	$\uparrow$ 0.00	$\Leftarrow$ 0.00	
	1	2	3	4	

- ▶ cost of 3 to move from striped cells (cost is 1 otherwise)
- ▶ moving from gray cells **unsuccessful** with probability 0.6

## Iterative Policy Evaluation for SSPs: Example

5	$\Rightarrow$ 1.00	$\Rightarrow$ 1.00	$\Rightarrow$ 1.00	$s_x$ 0.00	$\hat{V}_\pi^1$
4	$\Rightarrow$ 1.00	$\uparrow$ 1.00	$\uparrow$ 3.00	$\uparrow$ 1.00	
3	$\Rightarrow$ 1.00	$\uparrow$ 1.00	$\Leftarrow$ 1.00	$\Leftarrow$ 1.00	
2	$\uparrow$ 1.00	$\uparrow$ 1.00	$\uparrow$ 1.00	$\Leftarrow$ 1.00	
1	$\Rightarrow^{s_0}$ 1.00	$\Rightarrow$ 1.00	$\uparrow$ 1.00	$\Leftarrow$ 1.00	
	1	2	3	4	

- ▶ cost of 3 to move from striped cells (cost is 1 otherwise)
- ▶ moving from gray cells **unsuccessful** with probability 0.6

## Iterative Policy Evaluation for SSPs: Example

5	$\Rightarrow$ 2.00	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_x$ 0.00	$\hat{V}_\pi^2$
4	$\Rightarrow$ 2.00	$\uparrow$ 2.00	$\uparrow$ 5.20	$\uparrow$ 1.60	
3	$\Rightarrow$ 2.00	$\uparrow$ 2.00	$\Leftarrow$ 2.00	$\Leftarrow$ 2.00	
2	$\uparrow$ 2.00	$\uparrow$ 2.00	$\uparrow$ 2.00	$\Leftarrow$ 2.00	
1	$\Rightarrow^{s_0}$ 2.00	$\Rightarrow$ 2.00	$\uparrow$ 2.00	$\Leftarrow$ 2.00	
	1	2	3	4	

- ▶ cost of 3 to move from striped cells (cost is 1 otherwise)
- ▶ moving from gray cells **unsuccessful** with probability 0.6

## Iterative Policy Evaluation for SSPs: Example

5	$\Rightarrow$ 3.96	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_x$ 0.00	$\hat{V}_\pi^5$
4	$\Rightarrow$ 4.60	$\Uparrow$ 3.00	$\Uparrow$ 7.79	$\Uparrow$ 2.31	
3	$\Rightarrow$ 5.00	$\Uparrow$ 4.00	$\Leftarrow$ 5.00	$\Leftarrow$ 5.00	
2	$\Uparrow$ 5.00	$\Uparrow$ 5.00	$\Uparrow$ 5.00	$\Leftarrow$ 5.00	
1	$\Rightarrow^{s_0}$ 5.00	$\Rightarrow$ 5.00	$\Uparrow$ 5.00	$\Leftarrow$ 5.00	
	1	2	3	4	

- ▶ cost of 3 to move from striped cells (cost is 1 otherwise)
- ▶ moving from gray cells **unsuccessful** with probability 0.6

## Iterative Policy Evaluation for SSPs: Example

5	$\Rightarrow$ 4.46	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_x$ 0.00	$\hat{V}_\pi^{10}$
4	$\Rightarrow$ 5.43	$\Uparrow$ 3.00	$\Uparrow$ 8.44	$\Uparrow$ 2.50	
3	$\Rightarrow$ 6.38	$\Uparrow$ 4.00	$\Leftarrow$ 5.00	$\Leftarrow$ 7.31	
2	$\Uparrow$ 8.30	$\Uparrow$ 6.38	$\Uparrow$ 6.00	$\Leftarrow$ 8.18	
1	$\Rightarrow^{s_0}$ 9.00	$\Rightarrow$ 8.00	$\Uparrow$ 7.00	$\Leftarrow$ 8.96	
	1	2	3	4	

- ▶ cost of 3 to move from striped cells (cost is 1 otherwise)
- ▶ moving from gray cells **unsuccessful** with probability 0.6

## Iterative Policy Evaluation for SSPs: Example

5	$\Rightarrow$ 4.50	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_x$ 0.00	$\hat{V}_\pi^{29}$
4	$\Rightarrow$ 5.50	$\Uparrow$ 3.00	$\Uparrow$ 8.50	$\Uparrow$ 2.50	
3	$\Rightarrow$ 6.50	$\Uparrow$ 4.00	$\Leftarrow$ 5.00	$\Leftarrow$ 7.50	
2	$\Uparrow$ 9.00	$\Uparrow$ 6.50	$\Uparrow$ 6.00	$\Leftarrow$ 8.50	
1	$\Rightarrow^{s_0}$ 9.00	$\Rightarrow$ 8.00	$\Uparrow$ 7.00	$\Leftarrow$ 9.50	
	1	2	3	4	

- ▶ cost of 3 to move from striped cells (cost is 1 otherwise)
- ▶ moving from gray cells **unsuccessful** with probability 0.6

## Iterative Policy Evaluation: Algorithm

Iterative Policy Evaluation for SSP  $\mathcal{T}$ , policy  $\pi$  and  $\epsilon > 0$

initialize  $\hat{V}^0$  arbitrarily

**for**  $i = 1, 2, \dots$ :

**for all** states  $s \in S$ :

$$\hat{V}_\pi^i(s) := c(\pi(s)) + \sum_{s' \in S} T(s, \pi(s), s') \cdot \hat{V}_\pi^{i-1}(s')$$

**if**  $\max_{s \in S} |\hat{V}_\pi^i(s) - \hat{V}_\pi^{i-1}(s)| < \epsilon$ :

**return**  $\hat{V}_\pi^i$

## Iterative Policy Evaluation: Properties

### Theorem (Convergence of Iterative Policy Evaluation)

Let  $\mathcal{T}$  be an SSP,  $\pi$  be a proper policy for  $\mathcal{T}$  and  $\hat{V}_\pi^0(s) \in \mathbb{R}$  arbitrarily for all  $s \in S$ .

Iterative policy evaluation *converges* to the *true state-values*, i.e.,

$$\lim_{i \rightarrow \infty} \hat{V}_\pi^i(s) = V_\pi(s) \text{ for all } s \in S.$$

Proof omitted.

In practice, iterative policy evaluation converges to true state-values if  $\epsilon$  is small enough.

## Policy Evaluation: MDPs

What about *policy evaluation for MDPs*?

- ▶ MDPs (with finite state set) are *always cyclic*  
 $\Rightarrow$  backward induction not applicable
- ▶ but goal state *not required* for iterative policy evaluation
- ▶ albeit traces are infinite, iterative policy evaluation *converges*
- ▶ convergence theorem also holds for MDPs

## F3.3 Policy Improvement

## Example: Greedy Action

5	$\Rightarrow$ 4.50	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_*$ 0.00
4	$\Rightarrow$ 5.50	$\uparrow$ 3.00	$\uparrow$ 8.50	$\uparrow$ 2.50
3	$\Rightarrow$ 6.50	$\uparrow$ 4.00	$\Leftarrow$ 5.00	$\Leftarrow$ 7.50
2	$\uparrow$ 9.00	$\uparrow$ 6.50	$\uparrow$ 6.00	$\Leftarrow$ 8.50
1	$\Rightarrow^{s_0}$ 9.0	$\Rightarrow$ 8.00	$\uparrow$ 7.00	$\Leftarrow$ 9.50
	1	2	3	4

- ▶ Can we learn more from this than the state-values of a policy?



## Example: Greedy Action

5	$\Rightarrow$ 4.50	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_*$ 0.00
4	$\Rightarrow$ 5.50	$\uparrow$ 3.00	$\uparrow$ 8.50	$\uparrow$ 2.50
3	$\Rightarrow$ 6.50	$\uparrow$ 4.00	$\Leftarrow$ 5.00	$\uparrow$ 7.50
2	$\uparrow$ 9.00	$\uparrow$ 6.50	$\uparrow$ 6.00	$\Leftarrow$ 8.50
1	$\Rightarrow^{s_0}$ 9.0	$\uparrow$ 8.00	$\uparrow$ 7.00	$\Leftarrow$ 9.50
	1	2	3	4

- ▶ Can we learn more from this than the state-values of a policy?
- ▶ **Yes!** By evaluating all actions in each state, we can derive a **better policy**

## Greedy actions and policies

### Definition (Greedy Action)

Let  $s$  be a state of an SSP or MDP  $\mathcal{T}$  and  $V$  be a state-value function for  $\mathcal{T}$ . The **greedy action** in  $s$  with respect to  $V$  is

$$a_V(s) := \arg \min_{\ell \in L(s)} \left( c(\ell) + \sum_{s' \in S} T(s, \ell, s') \cdot V(s') \right).$$

The policy  $\pi_V$  with  $\pi_V(s) = a_V(s)$  is the **greedy policy**.

Determining the greedy policy of a given state-value function is called **policy improvement**.

## F3.4 Policy Iteration

## Policy Iteration

- ▶ Policy Iteration (PI) was first proposed by Howard in 1960
- ▶ exploits observation that **greedy actions** in result of policy evaluation describe **better** policy
- ▶ starts with arbitrary **policy**  $\pi_0$
- ▶ alternates **policy evaluation** and **policy improvement**
- ▶ as long as **policy changes**

## Example: Policy Iteration

5	$\Rightarrow$ 4.50	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_*$ 0.00	$\pi_0$
4	$\Rightarrow$ 5.50	$\uparrow$ 3.00	$\uparrow$ 8.50	$\uparrow$ 2.50	
3	$\Rightarrow$ 6.50	$\uparrow$ 4.00	$\Leftarrow$ 5.00	$\Leftarrow$ 7.50	
2	$\uparrow$ 9.00	$\uparrow$ 6.50	$\uparrow$ 6.00	$\Leftarrow$ 8.50	
1	$\Rightarrow^{s_0}$ 9.00	$\Rightarrow$ 8.00	$\uparrow$ 7.00	$\Leftarrow$ 9.50	
	1	2	3	4	

## Example: Policy Iteration

5	$\Rightarrow$ 4.50	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_*$ 0.00	$\pi_1$
4	$\Rightarrow$ 5.50	$\uparrow$ 3.00	$\uparrow$ 8.50	$\uparrow$ 2.50	
3	$\Rightarrow$ 6.50	$\uparrow$ 4.00	$\Leftarrow$ 5.00	$\uparrow$ 5.00	
2	$\uparrow$ 9.00	$\uparrow$ 6.50	$\uparrow$ 6.00	$\Leftarrow$ 8.50	
1	$\Rightarrow^{s_0}$ 8.50	$\uparrow$ 7.50	$\uparrow$ 7.00	$\Leftarrow$ 9.50	
	1	2	3	4	

## Example: Policy Iteration

5	$\Rightarrow$ 4.50	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_*$ 0.00	$\pi_2 = \pi_3$
4	$\Rightarrow$ 5.50	$\uparrow$ 3.00	$\uparrow$ 8.50	$\uparrow$ 2.50	
3	$\Rightarrow$ 6.50	$\uparrow$ 4.00	$\Leftarrow$ 5.00	$\uparrow$ 5.00	
2	$\uparrow$ 9.00	$\uparrow$ 6.50	$\uparrow$ 6.00	$\uparrow$ 7.50	
1	$\Rightarrow^{s_0}$ 8.50	$\uparrow$ 7.50	$\uparrow$ 7.00	$\Leftarrow$ 9.50	
	1	2	3	4	

## Policy Iteration: Algorithm

Policy Iteration for SSP or MDP  $\mathcal{T}$ 

initialize  $\pi_0$  to any policy (for SSP: proper)

**for**  $i = 0, 1, \dots$ :

  compute  $V_{\pi_i}$

  let  $\pi_{i+1}$  be the greedy policy w.r.t  $V_{\pi_i}$

**if**  $\pi_i = \pi_{i+1}$ :

**return**  $\pi_i$

## Properties

- ▶ PI computes **optimal policy** if policy evaluation is exact
- ▶ In practice, PI often requires **very few iterations** ...
- ▶ ... and is **much faster** than solving an LP

## F3.5 Summary

## Summary

- ▶ Policy evaluation for **acyclic policy** is possible in **one sweep** over the state space with **backward induction**
- ▶ **Iterative policy evaluation** applies state-value function iteratively and converges to true state-values
- ▶ **Greedy actions** in evaluated policy allow to **improve policy**
- ▶ **Policy iteration** alternates **policy evaluation** and **policy improvement**
- ▶ **Policy iteration** computes **optimal policy** (if policy evaluation is exact)