

# Planning and Optimization

## F4. Value Iteration

Malte Helmert and Gabriele Röger

Universität Basel

December 02, 2020

# Planning and Optimization

December 02, 2020 — F4. Value Iteration

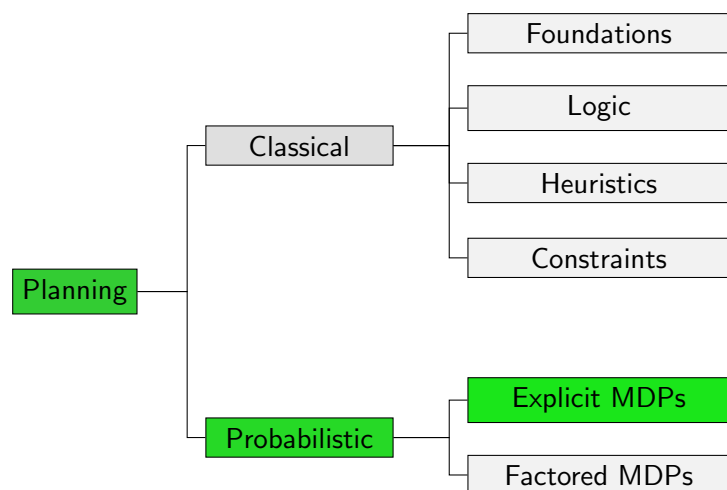
## F4.1 Introduction

## F4.2 Value Iteration

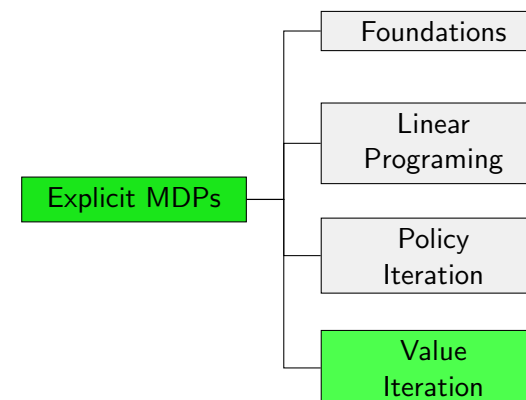
## F4.3 Asynchronous VI

## F4.4 Summary

## Content of this Course



## Content of this Course: Explicit MDPs



## F4.1 Introduction

## From Policy Iteration to Value Iteration

- ▶ Policy Iteration:
  - ▶ search over **policies**
  - ▶ by evaluating their **state-values**
- ▶ Value Iteration:
  - ▶ search directly over **state-values**
  - ▶ **optimal policy** induced by final state-values

## F4.2 Value Iteration

## Value Iteration: Idea

- ▶ Value Iteration (VI) was first proposed by Bellman in 1957
- ▶ computes estimates  $\hat{V}^0, \hat{V}^1, \dots$  of  $V_*$  in an **iterative** process
- ▶ starts with arbitrary  $\hat{V}^0$
- ▶ bases estimate  $\hat{V}^{i+1}$  on values of estimate  $\hat{V}^i$  by treating **Bellman equation** as **update rule** on all states:

$$\hat{V}^{i+1}(s) := \min_{a \in A(s)} \left( c(a) + \sum_{s' \in S} T(s, a, s') \cdot \hat{V}^i(s') \right)$$

(for SSPs; for MDPs accordingly)

- ▶ converges to state-values of **optimal policy**
- ▶ terminates when difference of estimates is small

## Example: Value Iteration

5	0.00	0.00	0.00	$s_*$ 0.00
4	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00
1	$s_0$ 0.00	0.00	0.00	0.00
	1	2	3	4

$\hat{V}^0$

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

## Example: Value Iteration

5	1.00	1.00	1.00	$s_*$ 0.00
4	1.00	1.00	3.00	1.00
3	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00
1	$s_0$ 1.00	1.00	1.00	1.00
	1	2	3	4

$\hat{V}^1$

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

## Example: Value Iteration

5	2.00	2.00	1.00	$s_*$ 0.00
4	2.00	2.00	5.20	1.60
3	2.00	2.00	2.00	2.00
2	2.00	2.00	2.00	2.00
1	$s_0$ 2.00	2.00	2.00	2.00
	1	2	3	4

$\hat{V}^2$

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

## Example: Value Iteration

5	3.96	2.00	1.00	$s_*$ 0.00
4	4.60	3.00	7.79	2.31
3	5.00	4.00	4.49	3.96
2	5.00	5.00	4.84	4.76
1	$s_0$ 5.00	5.00	5.00	4.97
	1	2	3	4

$\hat{V}^5$

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

## Example: Value Iteration

5	4.46	2.00	1.00	$s_*$	
4	5.43	3.00	8.44	2.48	
3	6.38	4.00	5.00	4.87	$\hat{V}^{10}$
2	8.30	6.38	6.00	6.95	
1	$s_0$	7.31	7.00	8.50	
	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

## Example: Value Iteration

5	4.50	2.00	1.00	$s_*$	
4	5.50	3.00	8.50	2.50	
3	6.50	4.00	5.00	5.00	$\hat{V}^{20}$
2	8.99	6.50	6.00	7.49	
1	$s_0$	7.50	7.00	9.49	
	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

## Example: Value Iteration

5	4.50	2.00	1.00	$s_*$	
4	5.50	3.00	8.50	2.50	
3	6.50	4.00	5.00	5.00	$\hat{V}^{29}$
2	9.00	6.50	6.00	7.50	
1	$s_0$	7.50	7.00	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

## Example: Value Iteration

5	$\Rightarrow$	$\Rightarrow$	$\Rightarrow$	$s_*$	
4	$\Rightarrow$	$\uparrow$	$\uparrow$	$\uparrow$	
3	$\Rightarrow$	$\uparrow$	$\Leftarrow$	$\uparrow$	$\pi_*$
2	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	
1	$\Rightarrow^{s_0}$	$\uparrow$	$\uparrow$	$\Leftarrow$	
	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

## Value Iteration for SSPs

Value Iteration for SSP  $\mathcal{T} = \langle S, A, c, T, s_0, S_* \rangle$  and  $\epsilon > 0$

initialize  $\hat{V}^0$  for 0 for goal states, otherwise arbitrarily

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

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

$$\hat{V}^{i+1}(s) := \min_{a \in A(s)} \left( c(a) + \sum_{s' \in S} T(s, a, s') \cdot \hat{V}^i(s') \right)$$

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

**return** a greedy policy  $\pi_{\hat{V}^{i+1}}$

## Value Iteration for MDPs

Value Iteration for MDP  $\mathcal{T} = \langle S, A, R, T, s_0, \gamma \rangle$  and  $\epsilon > 0$

initialize  $\hat{V}^0$  arbitrarily

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

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

$$\hat{V}^{i+1}(s) := \max_{a \in A(s)} \left( R(s) + \gamma \cdot \sum_{s' \in S} T(s, a, s') \cdot \hat{V}^i(s') \right)$$

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

**return**  $\pi_{\hat{V}^{i+1}}$

## F4.3 Asynchronous VI

### Asynchronous Value Iteration

- ▶ Updating all states simultaneously is called **synchronous backup**
  - ▶ Asynchronous VI performs backups for individual states
  - ▶ Different approaches lead to **different backup orders**
  - ▶ Can significantly **reduce computation**
  - ▶ **Guaranteed** to converge if all states are **selected repeatedly**
- ⇒ Optimal VI with **asynchronous backups** possible
- ⇒ No obvious termination criterion
- ⇒ often used in any-time setting (run until you need the result)

## In-place Value Iteration

- ▶ Synchronous value iteration creates new copy of value function (two are required simultaneously)

$$\hat{V}^{i+1}(s) := \min_{a \in A(s)} \left( c(a) + \sum_{s' \in S} T(s, a, s') \cdot \hat{V}^i(s') \right)$$

- ▶ In-place value iteration only requires a single copy of value function

$$\hat{V}(s) := \min_{a \in A(s)} \left( c(a) + \sum_{s' \in S} T(s, a, s') \cdot \hat{V}(s') \right)$$

- ▶ In-place VI is asynchronous because some backups are based on “old” values, some on “new” values

## F4.4 Summary

## Linear Programming, Policy Iteration, or Value Iteration?

- ▶ Linear Programming is the only technique where the solution is **guaranteed to be optimal** (independent from  $\epsilon$ )
- ▶ PI and VI are **often faster** than LP
- ▶ Policy evaluation is slightly cheaper than a VI iteration
  - ▶ PI faster than VI if **few iterations** required
  - ▶ VI faster than PI if number of PI iterations outweighs difference of policy evaluation compared to VI
- ▶ Asynchronous VI is basis of more sophisticated algorithm that can be applied in **large MDPs and SSPs**

## Summary

- ▶ Value Iteration searches in the **space of state-values**
- ▶ VI applies **Bellman equation** as update rule iteratively
- ▶ VI converges to **optimal** state-values
- ▶ VI **remains optimal** with **asynchronous backups** as long as all states are selected repeatedly