Planning and Optimization

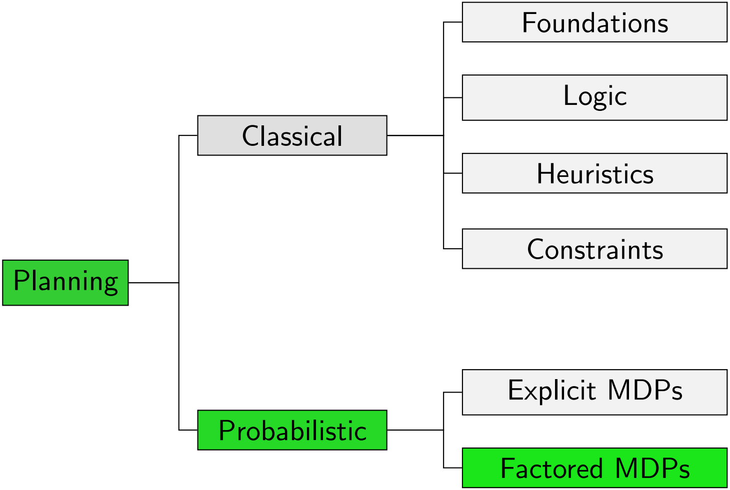
G3. Real-time Dynamic Programming

Malte Helmert and Thomas Keller

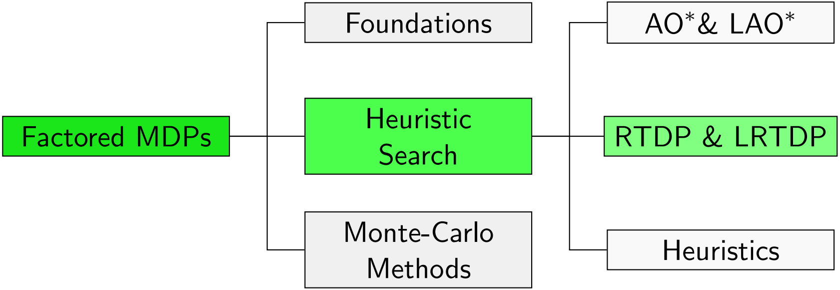
Universität Basel

December 9, 2019

Content of this Course



Content of this Course: Factored MDPs



Motivation

Comparison of Value Iteration and (L)AO*

Value Iteration and (L)AO* have different advantages:

- Both VI and (L)AO* compute optimal (executable) policy
- Admissible heuristic allows (L)AO* to restrict search to "relevant" part of the search space.
- VI operates on state table, no need to build an explicit representation of the search space
  (lower memory requirement for the same search space)

Real-time Dynamic Programming: Idea

Real-time Dynamic Programming (RTDP)
(Barto, Bradtke & Singh, 1995) combines these advantages:

- RTDP computes optimal (executable) policy
- RTDP uses an admissible heuristic to restrict search to "relevant" part of the search space
- RTDP operates on a state hash table that is built during seach

Real-time Dynamic Programming

Real-time Dynamic Programming

- RTDP updates only states relevant to the agent
- Originally motivated from agent that acts in environment by following greedy policy w.r.t. current state-value estimates.
- Performs Bellman backup in each encountered state
- Uses admissible heuristic for states not updated before

Trial-based Real-time Dynamic Programming

- We consider the offline version here.
  $\Rightarrow$ Interaction with environment is simulated in trials.
- In real world, outcome of action application cannot be chosen.
  $\Rightarrow$ In simulation, outcomes are sampled according to probabilities.

Real-time Dynamic Programming

RTDP for SSP $\mathcal{T}$

while more trials required:

$s := s_0$

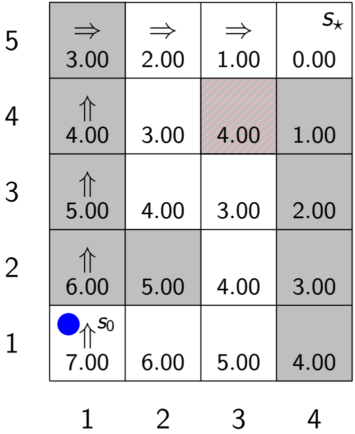
while $s \notin S_\star$:

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

$s :\sim \mathrm{succ}(s, a_{\hat{V}}(s))$

Note: $\hat{V}(s)$ is maintained as a hash table of states. On the right hand side of line 4 or 5, if a state s is not in $\hat{V}$, $h(s)$ is used.

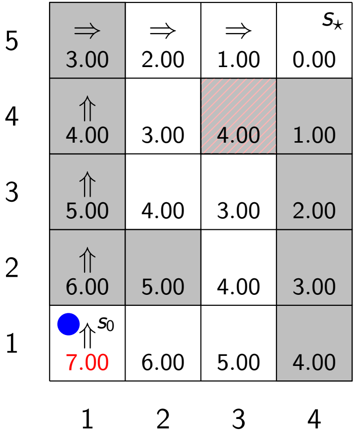
Example: RTDP



Start of 1st trial

Used heuristic: shortest path assuming agent never gets stuck

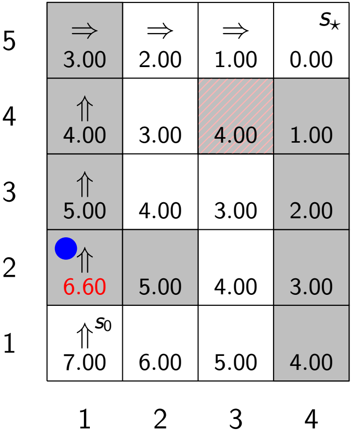
Example: RTDP



Step 1

Used heuristic: shortest path assuming agent never gets stuck

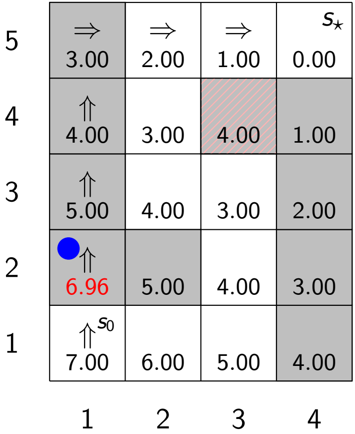
Example: RTDP



Step 2

Used heuristic: shortest path assuming agent never gets stuck

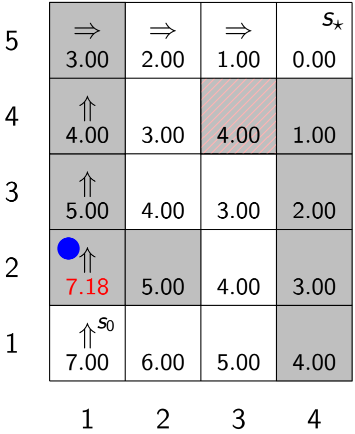
Example: RTDP



Step 3

Used heuristic: shortest path assuming agent never gets stuck

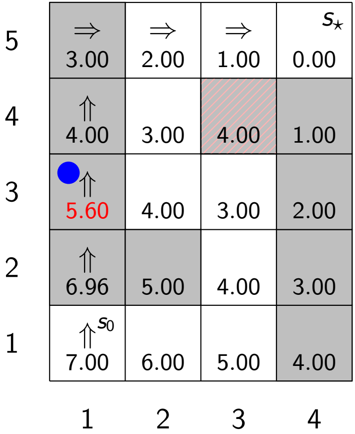
Example: RTDP



Step 4

Used heuristic: shortest path assuming agent never gets stuck

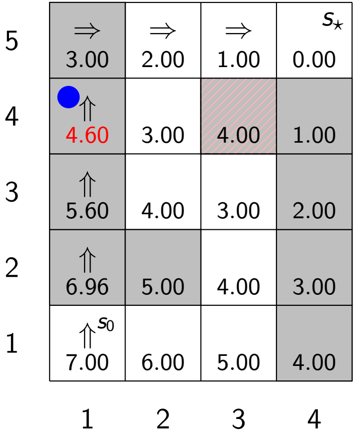
Example: RTDP



Step 5

Used heuristic: shortest path assuming agent never gets stuck

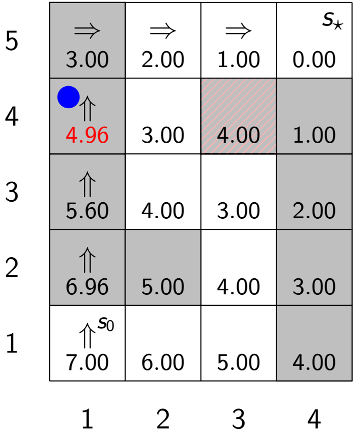
Example: RTDP



Step 6

Used heuristic: shortest path assuming agent never gets stuck

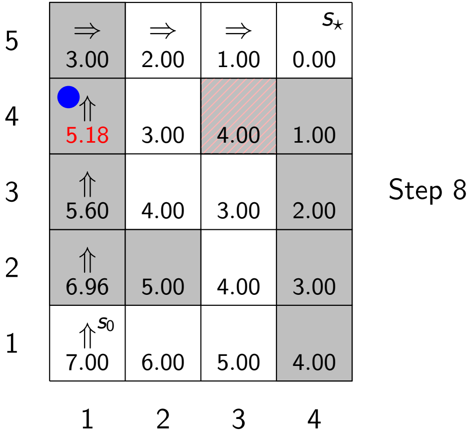
Example: RTDP



Step 7

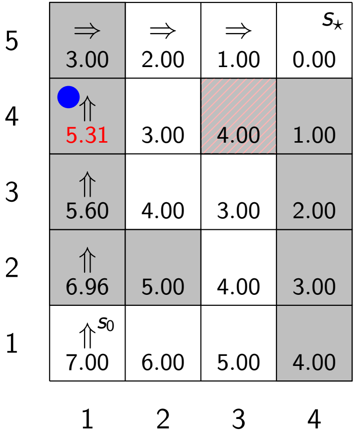
Used heuristic: shortest path assuming agent never gets stuck

Example: RTDP



Used heuristic: shortest path assuming agent never gets stuck

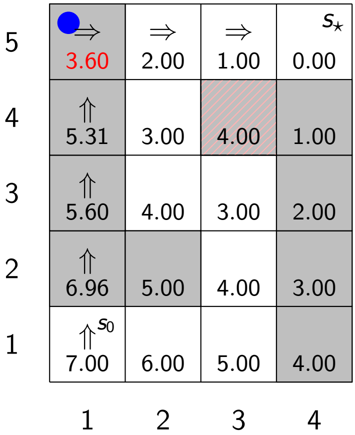
Example: RTDP



Step 9

Used heuristic: shortest path assuming agent never gets stuck

Example: RTDP



Step 10

Used heuristic: shortest path assuming agent never gets stuck

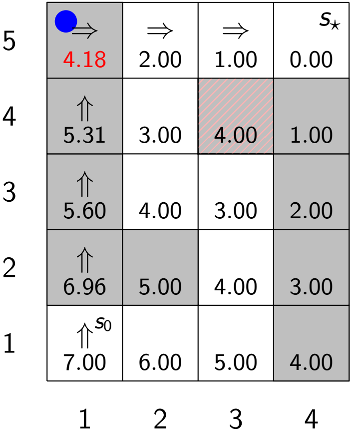
Example: RTDP

	1	2	3	4
5	$\Rightarrow$ 3.96	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_\star$ 0.00
4	$\Uparrow$ 5.31	3.00	4.00	1.00
3	$\Uparrow$ 5.60	4.00	3.00	2.00
2	$\Uparrow$ 6.96	5.00	4.00	3.00
1	s_0 $\Uparrow$ 7.00	6.00	5.00	4.00

Step 11

Used heuristic: shortest path assuming agent never gets stuck

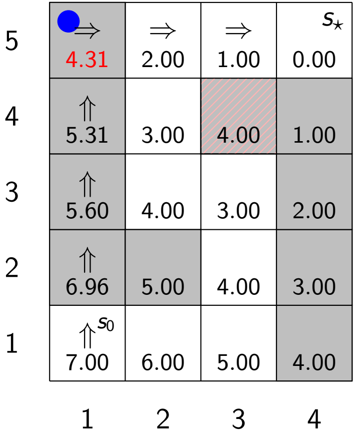
Example: RTDP



Step 12

Used heuristic: shortest path assuming agent never gets stuck

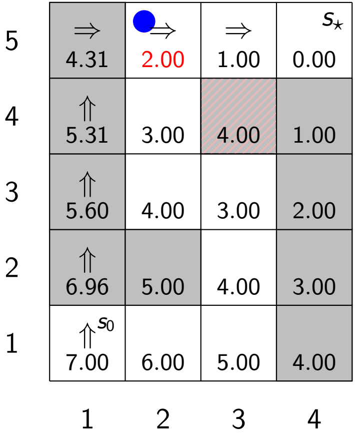
Example: RTDP



Step 13

Used heuristic: shortest path assuming agent never gets stuck

Example: RTDP



Step 14

Used heuristic: shortest path assuming agent never gets stuck

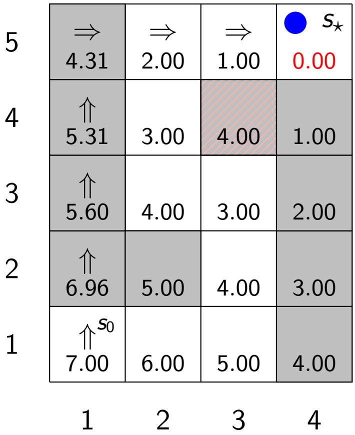
Example: RTDP

	1	2	3	4
5	$\Rightarrow$ 4.31	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_\star$ 0.00
4	$\Uparrow$ 5.31	3.00	4.00	1.00
3	$\Uparrow$ 5.60	4.00	3.00	2.00
2	$\Uparrow$ 6.96	5.00	4.00	3.00
1	$\Uparrow$ s_0 7.00	6.00	5.00	4.00

Step 15

Used heuristic: shortest path assuming agent never gets stuck

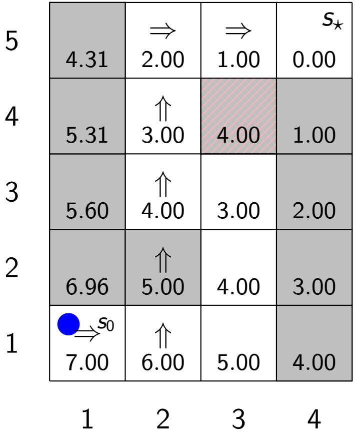
Example: RTDP



Step 16

Used heuristic: shortest path assuming agent never gets stuck

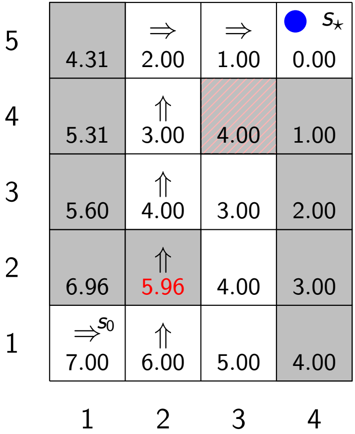
Example: RTDP



Start of 2nd trial

Used heuristic: shortest path assuming agent never gets stuck

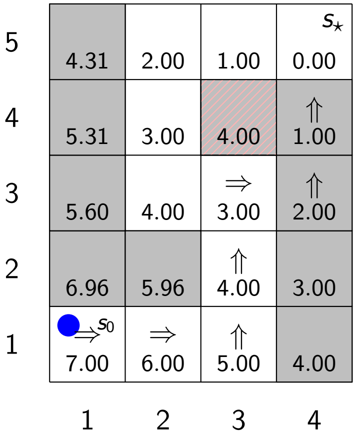
Example: RTDP



End of 2nd trial

Used heuristic: shortest path assuming agent never gets stuck

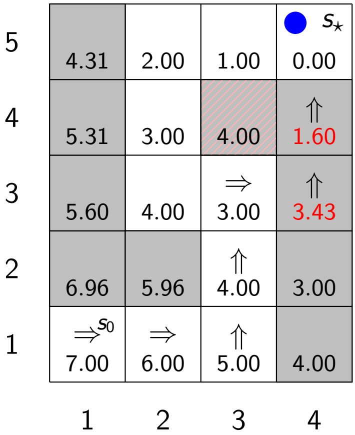
Example: RTDP



Start of 3rd trial

Used heuristic: shortest path assuming agent never gets stuck

Example: RTDP



End of 3rd trial

Used heuristic: shortest path assuming agent never gets stuck

Example: RTDP

5	4.31	$\Rightarrow$ 2.00	$\Rightarrow$ 1.00	$s_\star$ 0.00
4	5.31	$\Uparrow$ 3.00	7.92	2.38
3	6.18	$\Uparrow$ 4.00	5.00	4.80
2	7.77	$\Uparrow$ 6.50	6.00	7.03
1	$\Rightarrow$ s_0 8.50	$\Uparrow$ 7.50	7.00	7.18
	1	2	3	4

End of 16th trial

Used heuristic: shortest path assuming agent never gets stuck

RTDP: Theoretical Properties

Theorem

Using an admissible heuristic, RTDP converges to an optimal solution without (necessarily) computing state-value estimates for all states.

Proof omitted.

Labeled Real-time Dynamic Programming

Motivation

Issues of RTDP:

- States are still updated after state-value estimate has converged.
- No termination criterion $\Rightarrow$ algorithm is underspecified

Most popular algorithm to overcome these shortcomings:
Labeled RTDP (Bonet & Geffner, 2003)

Labeled RTDP: Idea

The main idea of Labeled RDTP (LRTDP) is to
label states as solved

- Each trial terminates when solved state is encountered
  $\Rightarrow$ solved states no longer updated
- LRTDP terminates when the initial state is labeled as solved
  $\Rightarrow$ well-defined termination criterion

Solved States in SSPs

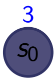
- States are solved if the state-value estimate changes only little
- In presence of cycles, all states in strongly connected component (SCC) are solved simultaneously
- Labeled RTDP uses sub-algorithm `CheckSolved` to check if all states in a SCC are solved

CheckSolved Procedure

- `CheckSolved` is called on all states that were encountered in a trial in reverse order.
- `CheckSolved` checks how much the state-value estimates of all states reachable under the greedy policy change and
- labels all those states as solved if the change is smaller than some constant ϵ.
- Otherwise, `CheckSolved` performs (additional) backup on reachable states for faster convergence.

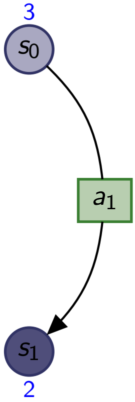
Labeled RTDP: Example ($\epsilon = 0.005$)

visited: s_0



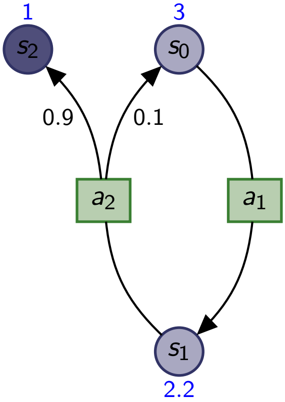
Labeled RTDP: Example ($\epsilon = 0.005$)

visited: s_0, s_1

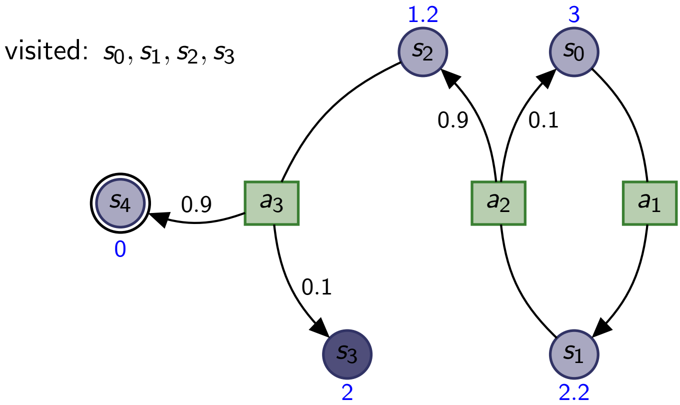


Labeled RTDP: Example ($\epsilon = 0.005$)

visited: s_0, s_1, s_2

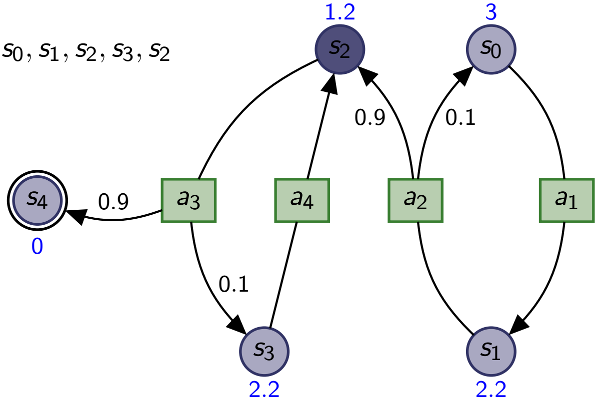


Labeled RTDP: Example ($\epsilon = 0.005$)



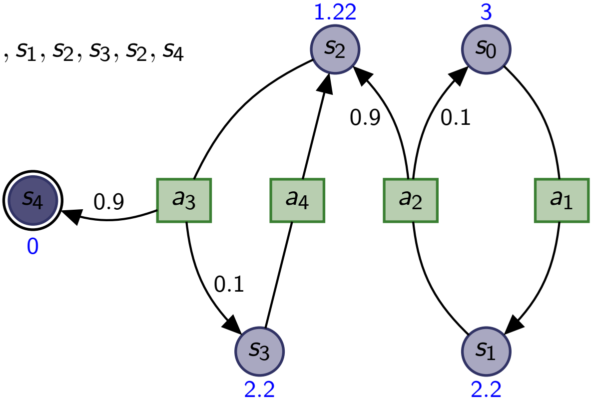
Labeled RTDP: Example ($\epsilon = 0.005$)

visited: s_0, s_1, s_2, s_3, s_2



Labeled RTDP: Example ($\epsilon = 0.005$)

visited: $s_0, s_1, s_2, s_3, s_2, s_4$

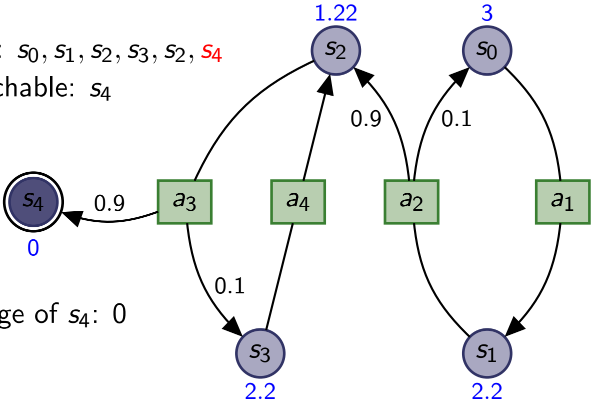


Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: s_4

change of s_4: 0



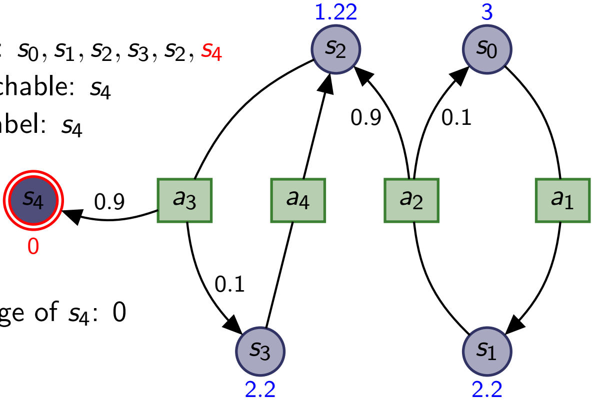
Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: s_4

label: s_4

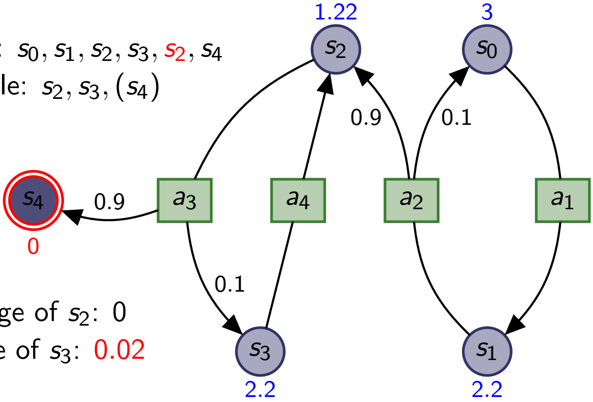
change of s_4: 0



Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: $s_2, s_3, (s_4)$



change of s_2: 0

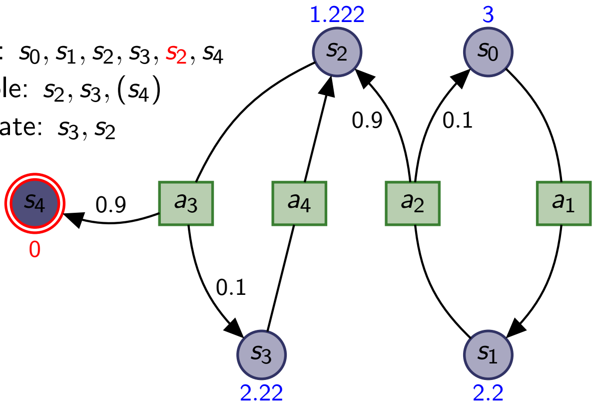
change of s_3: 0.02

Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: $s_2, s_3, (s_4)$

update: s_3, s_2



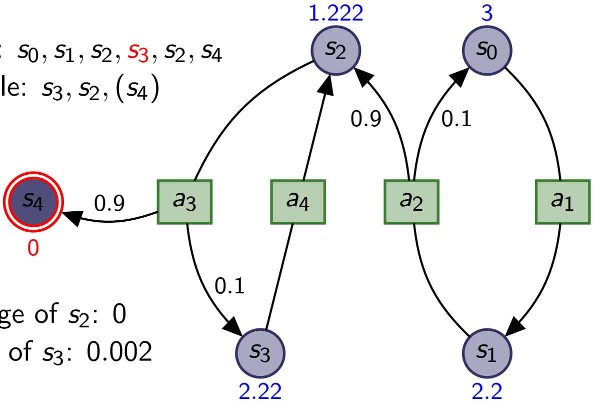
Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: $s_3, s_2, (s_4)$

change of s_2: 0

change of s_3: 0.002

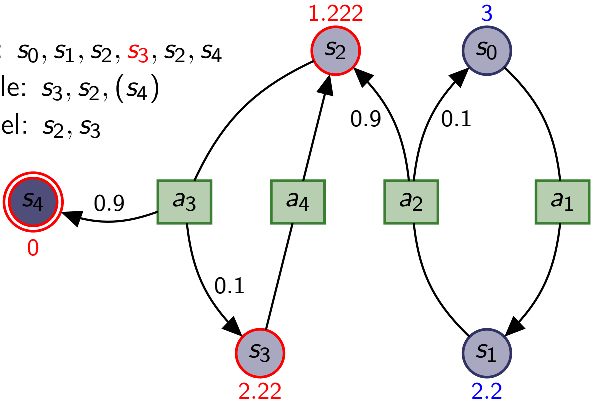


Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: $s_3, s_2, (s_4)$

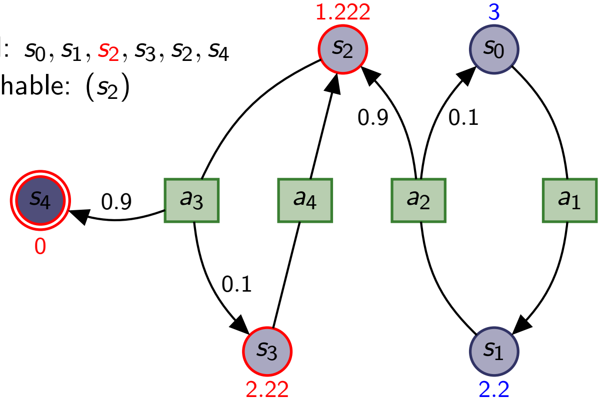
label: s_2, s_3



Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

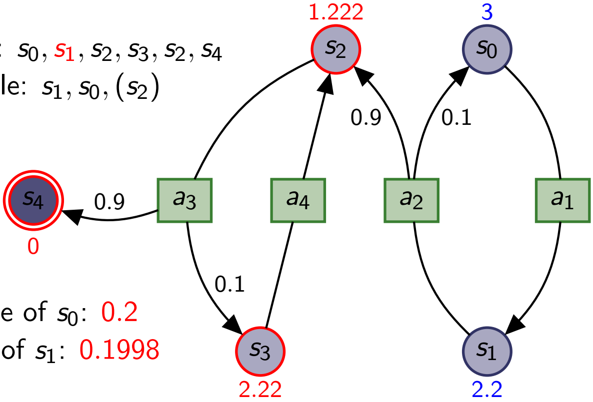
reachable: (s_2)



Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: $s_1, s_0, (s_2)$



change of s_0: 0.2

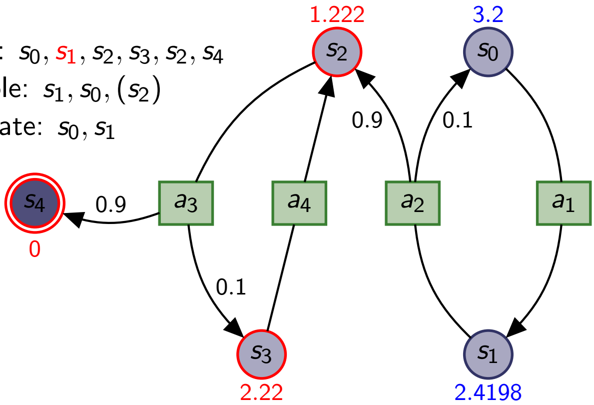
change of s_1: 0.1998

Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: $s_1, s_0, (s_2)$

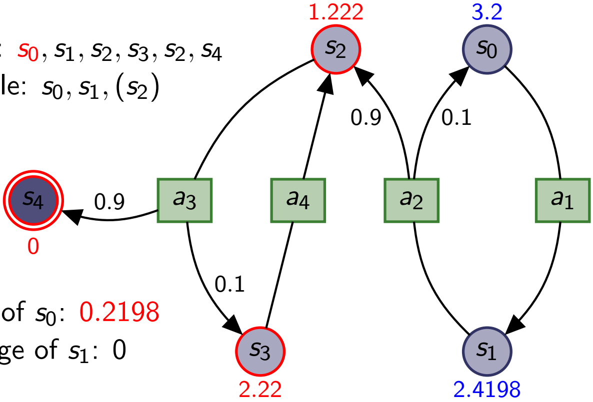
update: s_0, s_1



Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$
reachable: $s_0, s_1, (s_2)$

change of s_0: 0.2198
change of s_1: 0

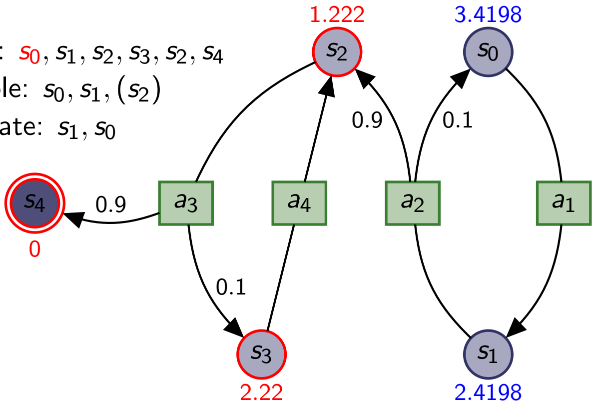


Labeled RTDP: Example ($\epsilon = 0.005$)

check_solved: $s_0, s_1, s_2, s_3, s_2, s_4$

reachable: $s_0, s_1, (s_2)$

update: s_1, s_0



Labeled Real-time Dynamic Programming

Labeled RTDP for SSP $\mathcal{T}$

while s_0 is not solved:
    visit(s_0)

visit state s

if s is solved or $s \in S_\star$:
    return
$\hat{V}(s) := \min_{\ell \in L(s)} \left(c(\ell) + \sum_{s' \in S} T(s, \ell, s') \cdot \hat{V}(s') \right)$
$s' :\sim \text{succ}(s, a_{\hat{V}}(s))$
visit(s')
check_solved(s)

$\hat{V}(s)$ is maintained as a hash table of states. On the right hand side of line 3 or 4 in `visit`(s), if a state s is not in $\hat{V}$, $h(s)$ is used.

Labeled RTDP: CheckSolved

check_solved for SSP $\mathcal{T}$

set ret := true, open, closed := stack
if s_0 not labeled **then** push $s0$ to open
while open is not empty:
    pop s from open and insert into closed
    if change of $s > \epsilon$
        ret := false
    else push all $s' \in \mathrm{succ}(s, a_{\hat{V}}(s))$ to open
        that are not labeled and not in open or closed
if ret **then** label all s in closed as solved
else perform backup on all s in closed

Labeled RTDP: Theoretical Properties

Theorem

Using an admissible heuristic, Labeled RTDP converges to an optimal solution without (necessarily) computing state-value estimates for all states.

Proof omitted.

Further RTDP Variants

Many variants exists, among them some interesting ones:

- Bounded RTDP (McMahan, Likhachev & Gordon, 2005)
- Focused RTDP (Smith & Simmons, 2006)
- Bayesian RTDP (Sanner et al., 2009)

Summary

Summary

- Real-time Dynamic Programming is an optimal algorithm for SSPs . . .
- . . . that backups only a subset of states . . .
- . . . without generating an explicit representation of the state-space.
- Labeled RTDP labels states as solved to stop updating converged states . . .
- . . . and speeds up convergence with additional backups in reverse order.