CSEP 573

Markov Decision Processes: Heuristic Search & Real-Time Dynamic Programming

Slides adapted from Andrey Kolobov and Mausam

1

Outline

- Stochastic Shortest Path (SSP) Problems
- Find-and Revise Framework
- Real-Time Dynamic Programming (RTDP)
- Heuristics
- LAO*

3

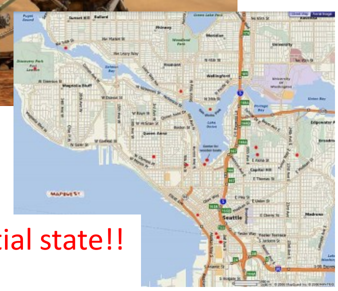
Stochastic Shortest-Path MDPs: Motivation

- **Assume the agent pays cost to achieve a goal**
- Example applications:
  - Controlling a Mars rover

    "How to collect scientific data without damaging the rover?"

  - Navigation

    "What's the fastest way to get to a destination, taking into account the traffic jams?"



Value & Policy Iteration don't represent initial state!!
Waste lots of effort!

10

Stochastic Shortest-Path MDPs: Definition

Bertsekas, 1995

SSP MDP is a tuple <*S, A, T, C, G*>, where:

- *S* is a finite state space, with a distinguished start state, s_0
- *A* is a finite action set
- *T*: *S* x *A* x *S* → [0, 1] is a stationary transition function
- *C*: *S* x *A* x *S* → $\mathbb{R}$ is a stationary *cost function* (low cost is good!)
- *G* is a set of absorbing cost-free goal states

Under two conditions:

- There is a *proper policy* (reaches a goal with P= 1 from all states)
- Every *improper policy* incurs a cost of ∞ from every state from which it does not reach the goal with P=1

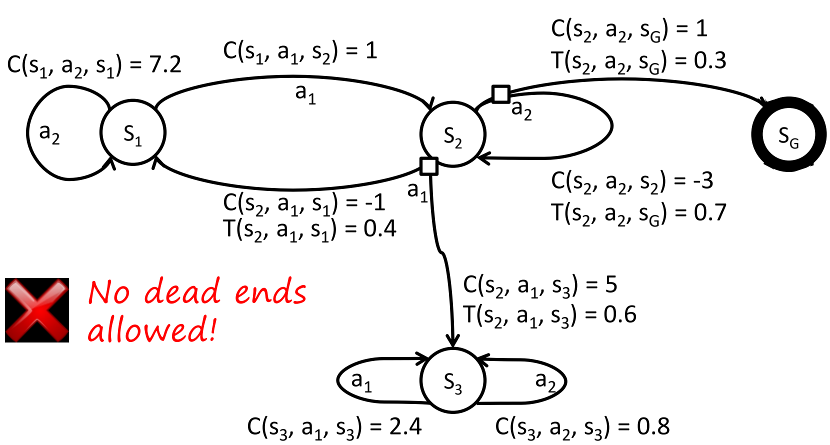
11

SSP MDP Details

- In SSP, *minimize* expected cost
- Every cost-minimizing policy is proper!
- **Thus, an optimal policy = cheapest way to a goal**
- Why are SSP MDPs called “indefinite-horizon”?
  - If a policy is optimal, it will take a finite, but apriori unknown, time to reach goal

12

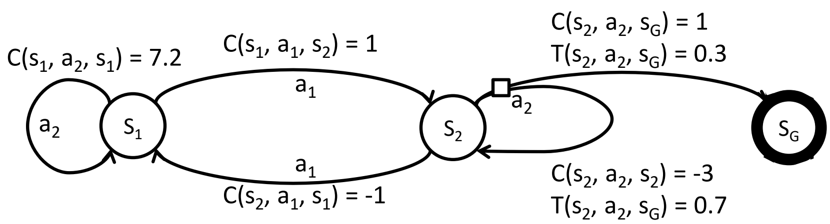
SSP MDP Example, not!



No dead ends allowed!

13

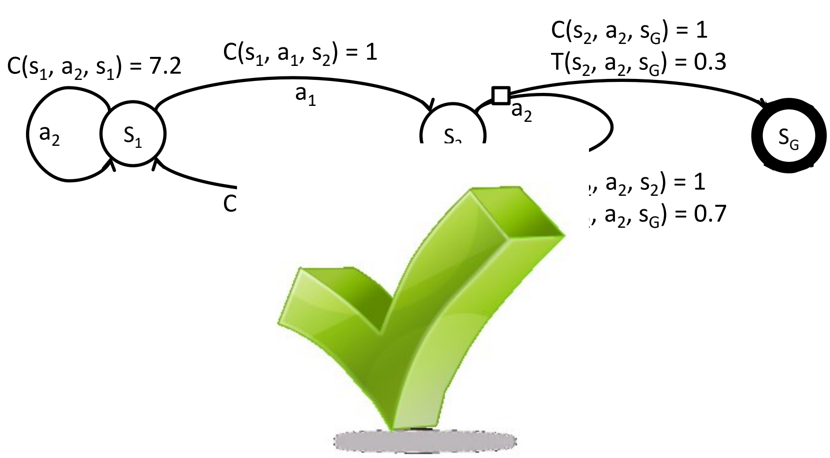
SSP MDP Example, also not!



No cost-free "loops" allowed!

14

SSP MDP Example



15

SSP MDPs: Optimality Principle

For an SSP MDP, let:

Note: no discounting!

Exp. Lin. Add. Utility

– $V^\pi(h) = \mathbb{E}_h^\pi[C_1 + C_2 + \ldots]$ for all h

For every history, the value of a policy is well-defined!

Then:

Every policy either takes a finite exp. # of steps to reach a goal, or has an infinite cost.

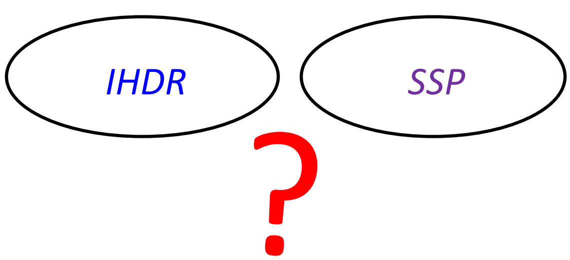
– V^* exists and is stationary Markovian, π^* exists and is stationary deterministic Markovian
– For all s:

$$\boldsymbol{V^*(s) = \min_{a \text{ in } A} \left[\sum_{s' \text{ in } S} T(s, a, s') \left[C(s, a, s') + V^*(s') \right] \right]}$$
$$\boldsymbol{\pi^*(s) = \operatorname{argmin}_{a \text{ in } A} \left[\sum_{s' \text{ in } S} T(s, a, s') \left[C(s, a, s') + V^*(s') \right] \right]}$$

16

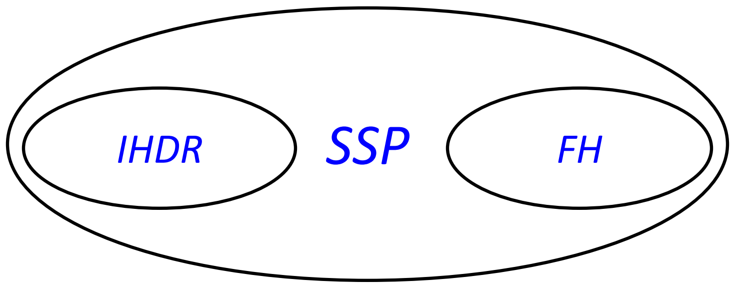
SSP and Other MDP Classes

E.g., Indefinite-horizon discounted reward



18

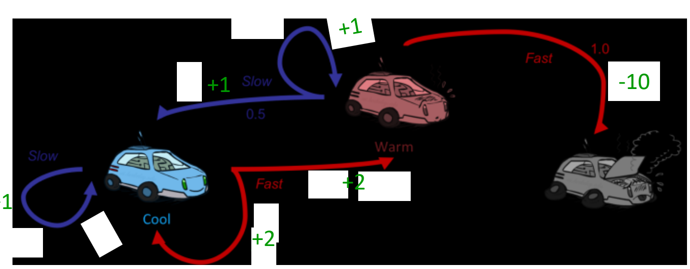
SSP and Other MDP Classes



- ***FH* => *SSP***: ∀s, ∀i ∈[0, L] create a new states (s, i); (s,0) are goals
- ***IHDR* => *SSP***: add γ-probability transitions to goal
- **Will concentrate on *SSP* in the rest of the tutorial**

19

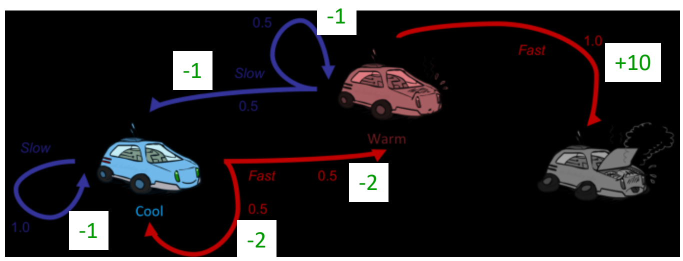
IHDR → SSP



20

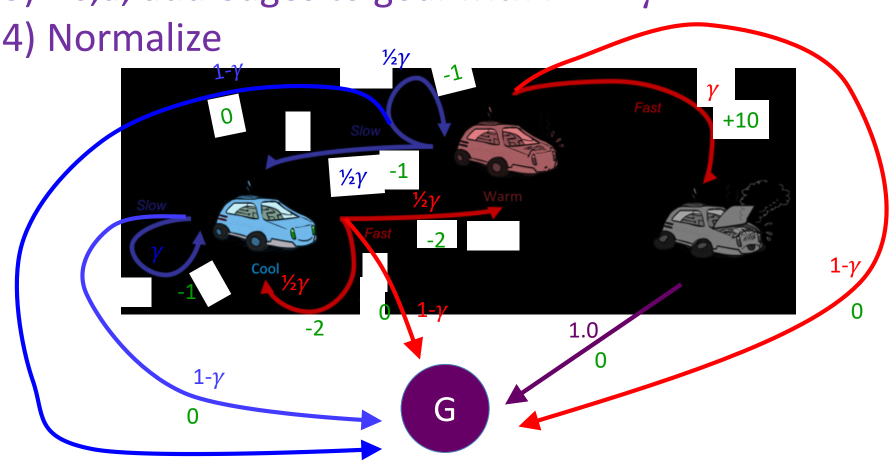
IHDR → SSP

1) Invert rewards to costs



IHDR → SSP

1) Invert rewards to costs
2) Add new goal state & edges from absorbing states
3) ∀s,a, add edges to goal with P = 1-γ
4) Normalize



Computational Complexity of MDPs

- **Good news**:
  - Solving *IHDR, SSP* in flat representation is *P*-complete
  - Solving *FH* in flat representation is *P*-hard
  - That is, they don't benefit from parallelization, but are solvable in polynomial time!

24

Computational Complexity of MDPs

- **Bad news**:
  - Solving *FH, IHDR, SSP* in factored representation is *EXPTIME*-complete!
  - Flat representation doesn't make MDPs harder to solve, it makes big ones easier to describe.

25

(General) Asynchronous VI

initialize V arbitrarily for each state
while $Res^V > \epsilon$ **do**
    select a state s
    compute $V(s)$ using a Bellman backup at s
    update $Res^V(s)$
end
return greedy policy π^V

$Res^V(s) = |\ V(s) - \max \sum T(s,a,s')[R(s,a,s')+V(s')]\ |$
$Res^V = \max_s Res^V(s)$

Heuristic Search

- Insight 1
  - knowledge of a start state, s_0, to save on computation
    - ~ (all sources shortest path → single source shortest path)
- Insight 2
  - additional knowledge in the form of heuristic function
    - ~ (dfs/bfs → A*)

42

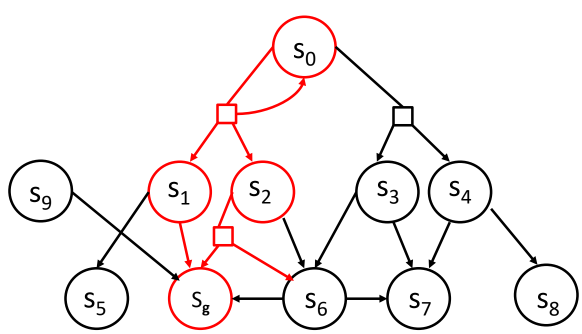
Partial Policy

- Define *Partial policy*
  - π: *S'* → *A*, where *S'* ⊆ *S*

- Define *Partial policy closed w.r.t. a state s.*
  - is a partial policy π_s
  - defined for all states *s'* reachable by π_s starting from *s*

44

Partial policy closed wrt s_0



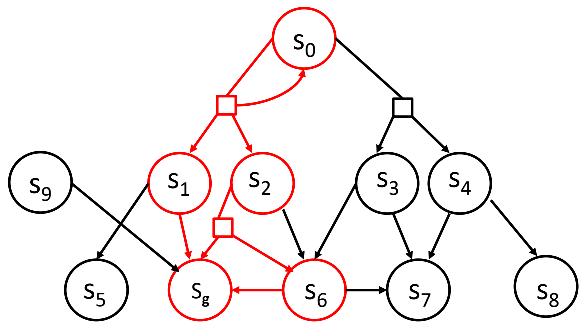
a_1 is left action
a_2 is on right

Is this policy closed wrt s_0?

$\pi_{s0}(s_0)= a_1$
$\pi_{s0}(s_1)= a_2$
$\pi_{s0}(s_2)= a_1$

45

Partial policy closed wrt s_0



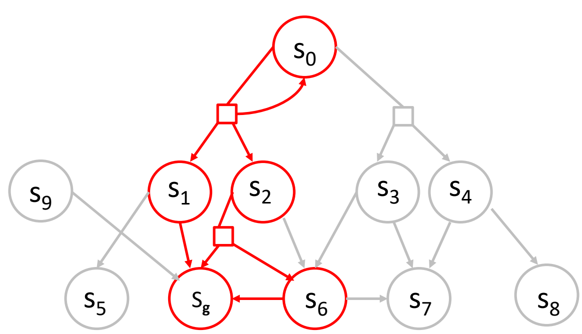
a_1 is left action
a_2 is on right

Is this policy closed wrt s_0?

$\pi_{s0}(s_0)= a_1$
$\pi_{s0}(s_1)= a_2$
$\pi_{s0}(s_2)= a_1$
$\pi_{s0}(s_6)= a_1$

46

Policy Graph of π_{s0}



a_1 is left action
a_2 is on right

$\pi_{s0}(s_0)= a_1$
$\pi_{s0}(s_1)= a_2$
$\pi_{s0}(s_2)= a_1$
$\pi_{s0}(s_6)= a_1$

47

Greedy Policy Graph

- Define *greedy policy*: $\pi^V = \text{argmin}_a Q^V(s,a)$
- Define *greedy partial policy rooted at s_0*
  - Partial policy rooted at s_0
  - Greedy policy
  - denoted by π^V_{s0}
- Define *greedy policy graph*
  - Policy graph of π^V_{s0} : denoted by G^V_{s0}

48

Heuristic Function

- h(s): S→$\mathbb{R}$
  - estimates V*(s)
  - gives an indication about "goodness" of a state
  - usually used in initialization $V_0(s) = h(s)$
  - helps us avoid seemingly bad states
- Define *admissible* heuristic
  - Optimistic (underestimates cost)
  - $h(s) \leq V^*(s)$

49

A General Scheme for Heuristic Search in MDPs

- Two (over)simplified intuitions
  - Focus on states in greedy policy *wrt.* V rooted at s_0
  - Focus on states with residual > ε
- Find & Revise:
  - repeat
    - find a state that satisfies the two properties above
    - perform a Bellman backup
  - until no such state remains

FIND & REVISE [Bonet&Geffner 03a]

Start with a heuristic value function $V \leftarrow h$
while *V's greedy graph $G^V_{s_0}$ contains a state s with $Res^V(s) > \epsilon$* **do**
    FIND a state s in $G^V_{s_0}$ with $Res^V(s) > \epsilon$
    REVISE $V(s)$
end
return a π^V

(perform Bellman backups)

- Convergence to V* is guaranteed
  - if heuristic function is admissible, and
  - ~no state gets starved in ∞ FIND steps

Heuristic Search Algorithms

- Definitions
- Find & Revise Scheme.
- LAO* and Extensions
- RTDP and Extensions
- Other uses of Heuristics/Bounds
- Heuristic Design

89

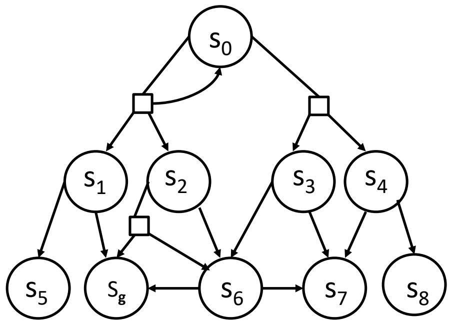
Real Time Dynamic Programming

[Barto et al 95]

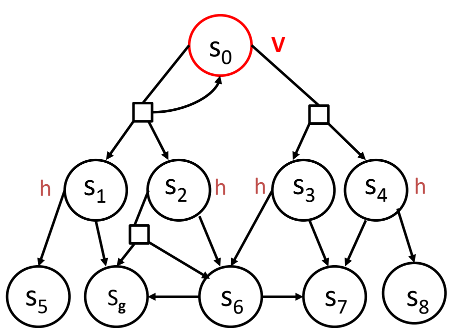
- Original Motivation
  - agent acting in the real world
- Trial
  - simulate greedy policy starting from start state;
  - perform Bellman backup on visited states
  - stop when you hit the goal
- RTDP: repeat trials forever
  - Converges in the limit #trials → ∞

90

Trial



Trial



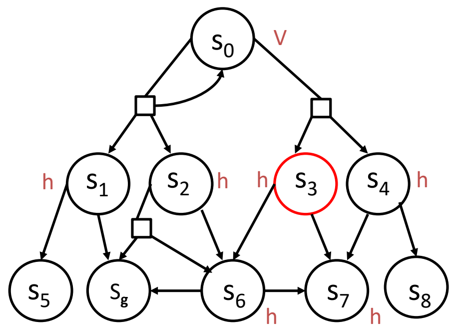
start at start state

repeat

perform a Bellman backup

simulate greedy action

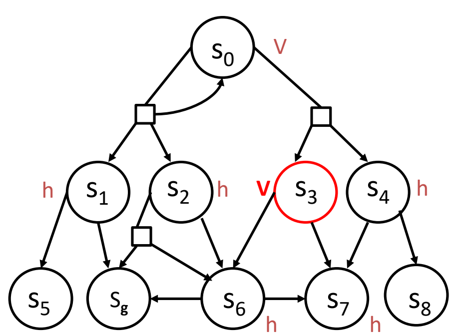
Trial



start at start state
repeat
perform a Bellman backup
simulate greedy action

93

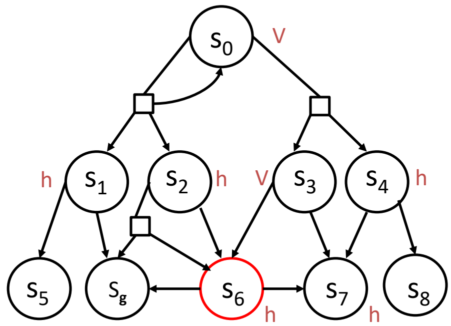
Trial



start at start state
repeat
perform a Bellman backup
simulate greedy action

94

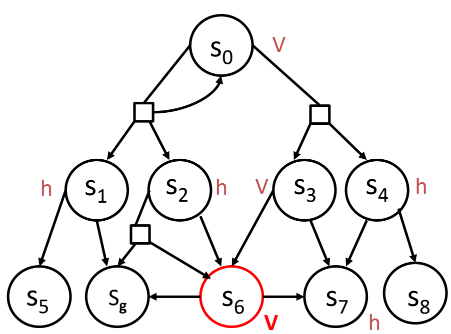
Trial



start at start state
repeat
perform a Bellman backup
simulate greedy action

95

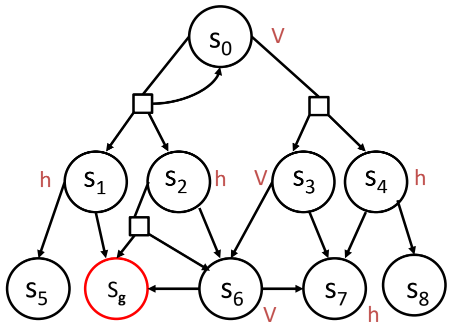
Trial



start at start state
repeat
perform a Bellman backup
simulate greedy action

96

Trial

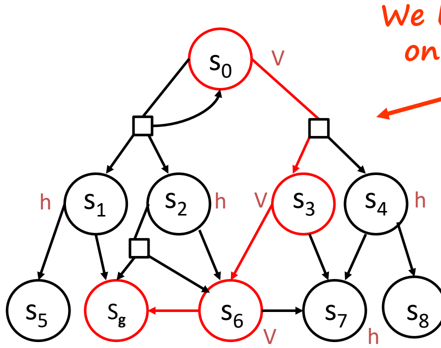


start at start state
repeat
perform a Bellman backup
simulate greedy action
until hit the goal

97

Trial

We backed-up all states on trajectory to goal



RTDP

repeat forever

start at start state
repeat
perform a Bellman backup
simulate greedy action
until hit the goal

98

Real Time Dynamic Programming

[Barto et al 95]

Ignores residual; Lacks focus!

Spot any problems?

- Trial
  - simulate greedy policy starting from start state;
  - perform Bellman backup on visited states
  - stop when you hit the goal

No termination condition!

- RTDP: repeat trials forever
  - Converges in the limit #trials → ∞

99

RTDP Family of Algorithms

repeat
    $s \leftarrow s_0$

    repeat *//trials*
        REVISE s; identify a_{greedy}
        FIND: pick s' s.t. $T(s, a_{greedy}, s') > 0$
        $s \leftarrow s'$
    until $s \in G$

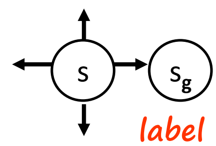
until termination test

100

Termination Test Take 1: Labeling

- Admissible heuristic & monotonicity
  - ⇒ $V(s) \le V^*(s)$
  - ⇒ $Q(s,a) \le Q^*(s,a)$
- If V(s) has converged
  - Then label state, s, as solved

Case 0

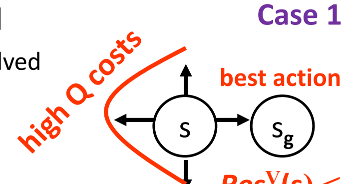


label all goals as solved

Termination Test Take 1: Labeling

- Admissible heuristic & monotonicity
  - ⇒ $V(s) \le V^*(s)$
  - ⇒ $Q(s,a) \le Q^*(s,a)$
- If V(s) has converged
  - Then label state, s, as solved

Case 1

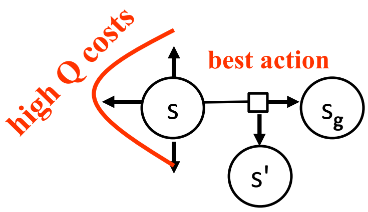


$Res^V(s) < \varepsilon$

⇒ *V(s) won't change!*

label s as solved

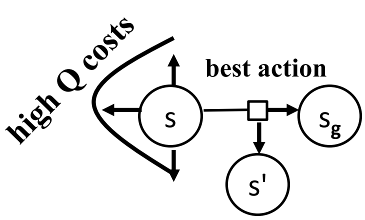
Labeling (contd)



$Res^V(s) < \varepsilon$
s' already solved
$\Rightarrow$ V(s) won't change!

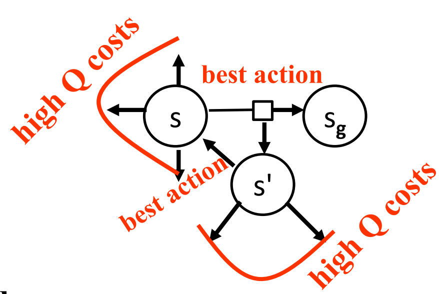
label s as solved

Labeling (contd)



$Res^V(s) < \varepsilon$
s' already solved
$\Rightarrow$ V(s) won't change!

label s as solved



$Res^V(s) < \varepsilon$
$Res^V(s') < \varepsilon$

V(s), V(s') won't change!
label s, s' as solved

Labeled RTDP [Bonet&Geffner 03b]

repeat
    $s \leftarrow s_0$
    label all goal states as solved

    repeat *//trials*
        REVISE s; identify a_{greedy}
        FIND: sample s' from T(s, a_{greedy}, s')
        $s \leftarrow s'$
    until s is solved

    for all states s in the trial
        try to label s as solved
until s_0 is solved

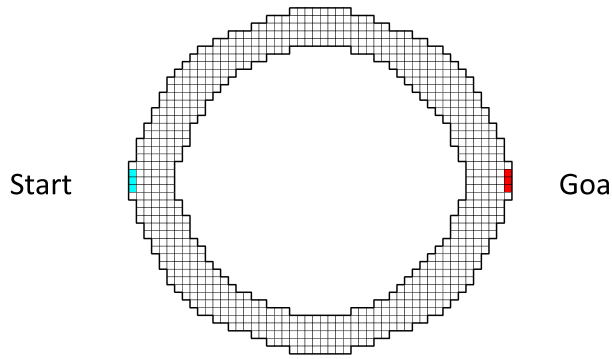
105

LRTDP

- terminates in finite time
  – due to labeling procedure
- anytime
  – focuses attention on more probable states
- fast convergence
  – focuses attention on unconverged states

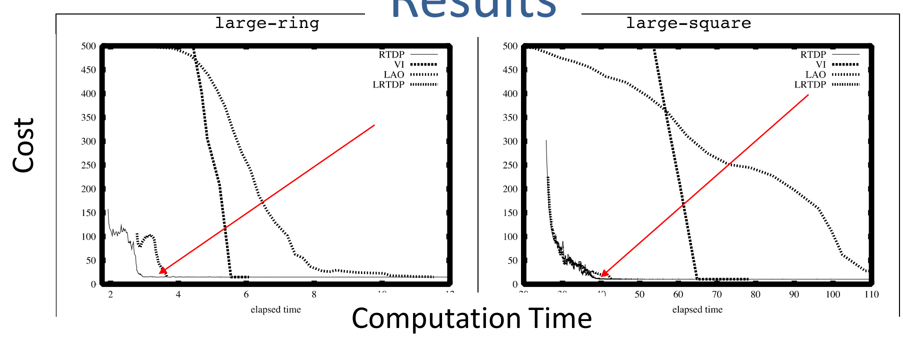
106

LRTDP Experiments



Racetrack Domain

Results



h=0

algorithm	small-b	large-b	h-track	small-r	large-r	small-s	large-s	small-y	large-y
VI($h = 0$)	1.101	**4.045**	**15.451**	0.662	5.435	5.896	78.720	16.418	61.773
ILAO*($h = 0$)	2.568	11.794	43.591	1.114	11.166	12.212	250.739	57.488	182.649
LRTDP($h = 0$)	**0.885**	7.116	15.591	**0.431**	**4.275**	**3.238**	**49.312**	**9.393**	**34.100**

Table 2: Convergence time in seconds for the different algorithms with initial value function $h = 0$ and $\epsilon = 10^{-3}$. Times for RTDP not shown as they exceed the cutoff time for convergence (10 minutes). Faster times are shown in bold font.

$h=h_{min}$

algorithm	small-b	large-b	h-track	small-r	large-r	small-s	large-s	small-y	large-y
VI(h_{min})	1.317	4.093	12.693	0.737	5.932	6.855	102.946	17.636	66.253
ILAO*(h_{min})	1.161	2.910	11.401	0.309	3.514	0.387	1.055	0.692	1.367
LRTDP(h_{min})	**0.521**	**2.660**	**7.944**	**0.187**	**1.599**	**0.259**	**0.653**	**0.336**	**0.749**

Table 3: Convergence time in seconds for the different algorithms with initial value function $h = h_{min}$ and $\epsilon = 10^{-3}$. Times for RTDP not shown as they exceed the cutoff time for convergence (10 minutes). Faster times are shown in bold font.

Picking a Successor Take 2

- Labeled RTDP/RTDP: sample $s' \propto T(s, a_{greedy}, s')$
  - Advantages
    - more probable states are explored first
    - no time wasted on converged states
  - Disadvantages
    - Convergence test is a hard constraint
    - Sampling ignores “amount” of convergence
- If we knew how much V(s) was expected to change?
  - sample $s' \propto$ expected change

109

Upper Bounds in SSPs

- RTDP/LAO* maintain lower bounds
  - call it V_l
- Additionally associate upper bound with s
  - $V_u(s) \geq V^*(s)$
- Define $gap(s) = V_u(s) – V_l(s)$
  - low gap(s): more converged a state
  - high gap(s): more expected change in its value

110

Backups on Bounds

- Recall monotonicity
- Backups on lower bound
  - continue to be lower bounds
- Backups on upper bound
  - continues to be upper bounds
- Intuitively
  - V_l will increase to converge to V*
  - V_u will decrease to converge to V*

111

Bounded RTDP [McMahan et al 05]

repeat
    $s \leftarrow s_0$

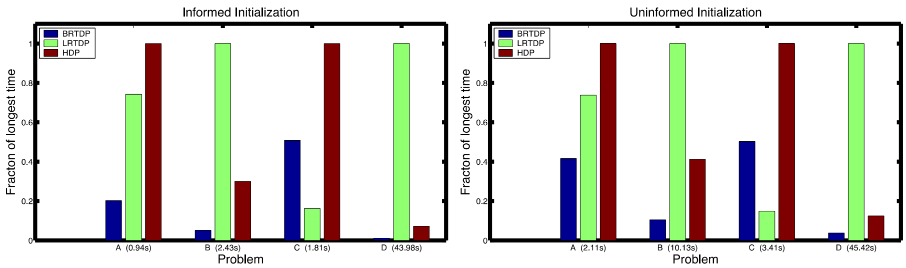
    repeat *//trials*
        identify a_{greedy} based on V_l
        FIND: sample $s' \propto T(s, a_{greedy}, s').gap(s')$
        $s \leftarrow s'$
    until $gap(s) < \varepsilon$

    for all states s in trial in reverse order
        REVISE s // backup both upper and lower bounds

until $gap(s_0) < \varepsilon$

112

BRTDP Results



A, B – racetrack; C,D gridworld. A,C have sparse noise; B,D much noise

113