CSE 573

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

Slides adapted from Andrey Kolobov and Mausam

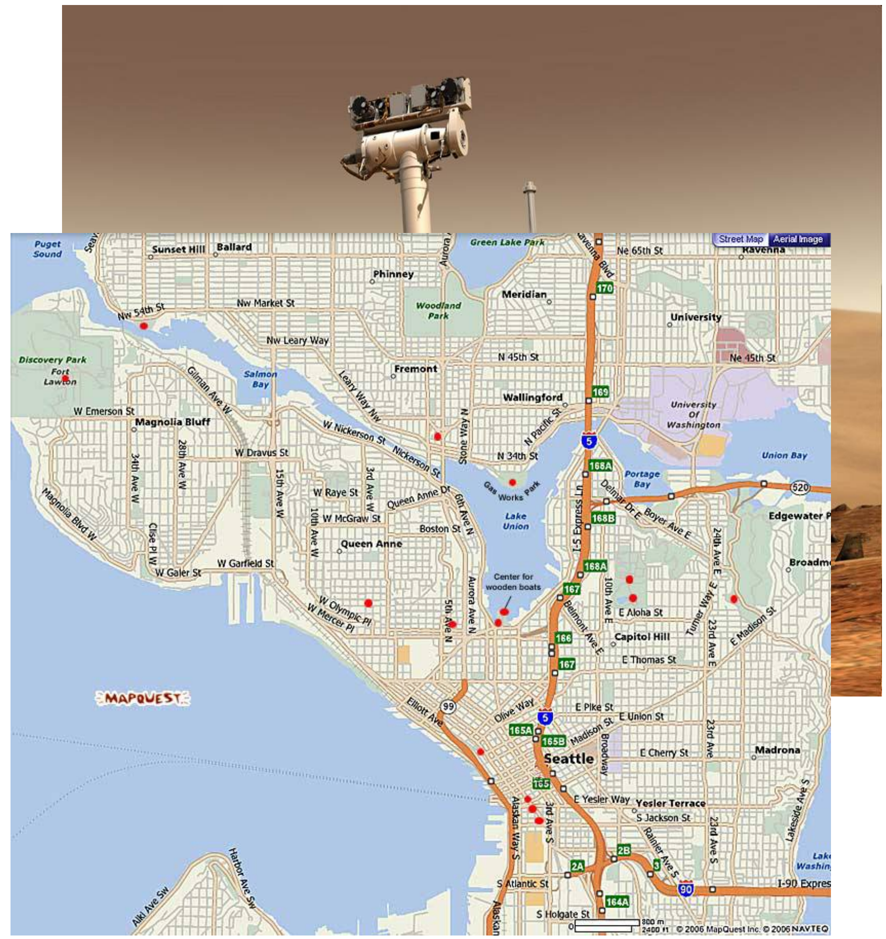
Stochastic Shortest-Path MDPs: Motivation

- **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?"



Stochastic Shortest-Path MDPs: Definition

Bertsekas, 1995

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

- *S* is a finite state space
- (*D* is an infinite sequence *(1,2, …)*)
- *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

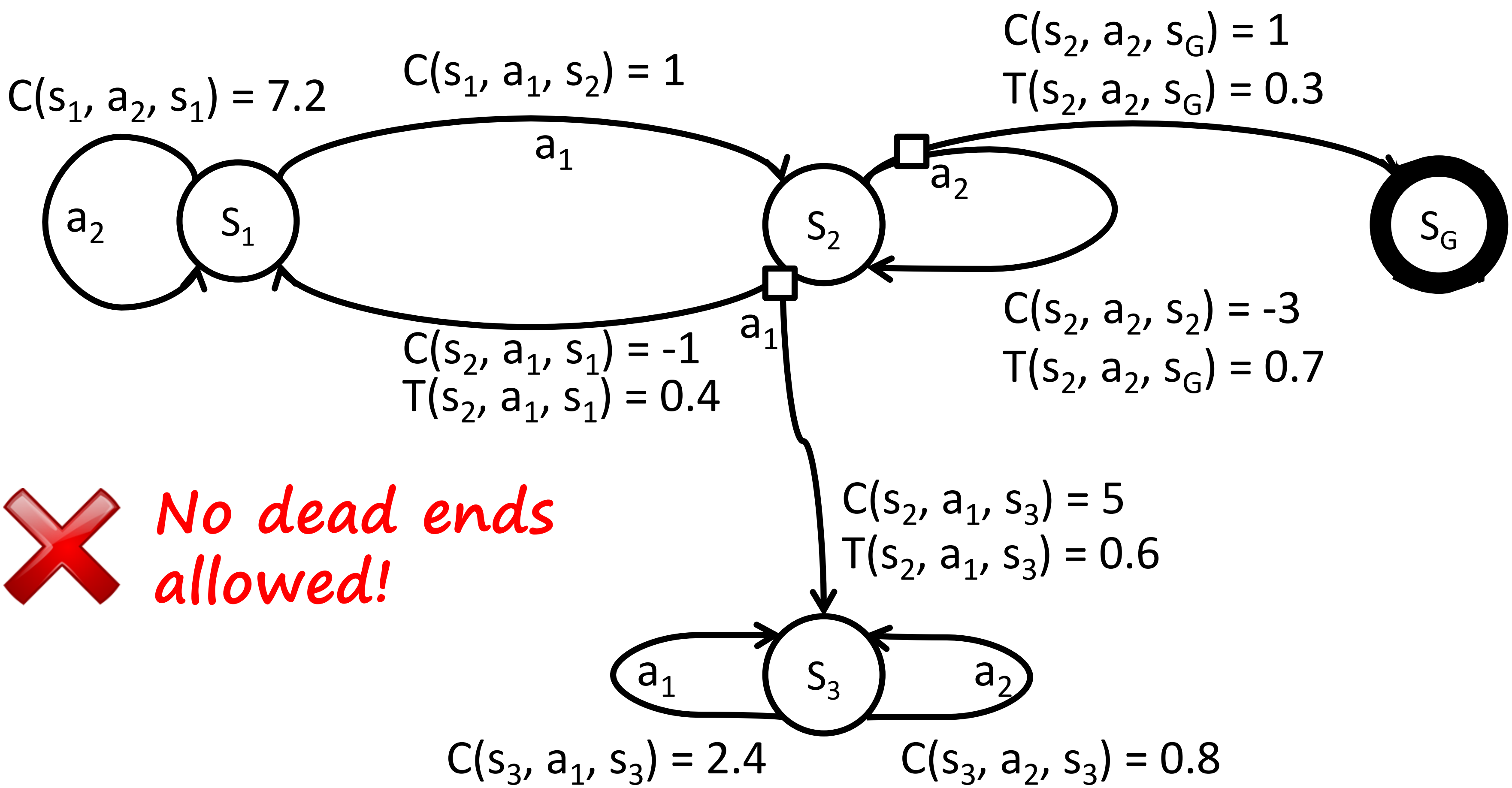
Under two conditions:

- 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

SSP MDP Details

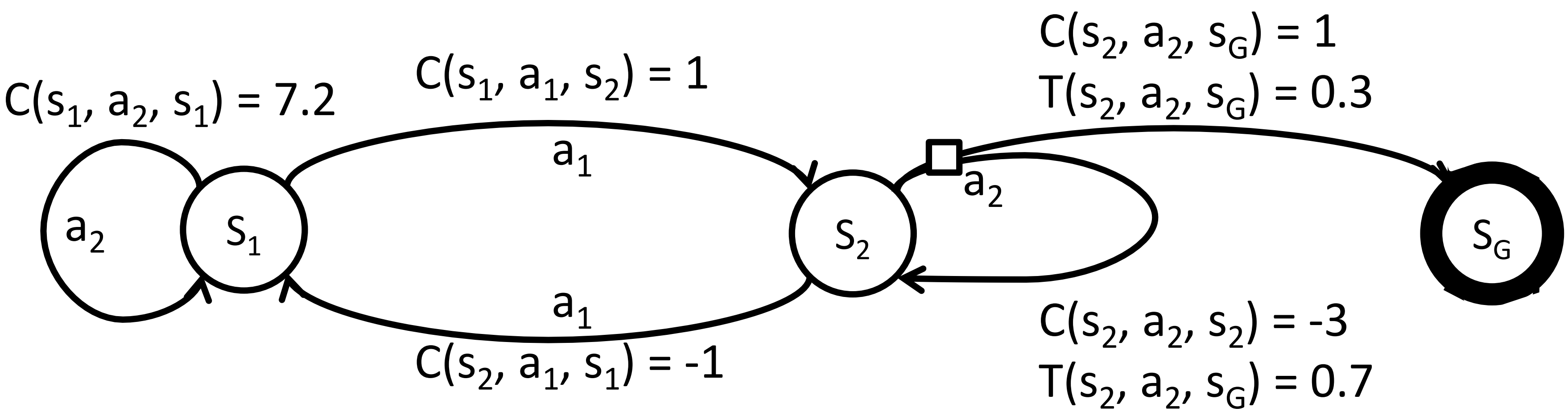
- In SSP, maximizing ELAU = *minimizing* exp. cost
- Every cost-minimizing policy is proper!
- **Thus, an optimal policy = cheapest way to a goal**
- Why are SSP MDPs called "indefinite-horizon"?
  - If a policy is optimal, it will take a finite, but apriori unknown, time to reach goal

SSP MDP Example, not!



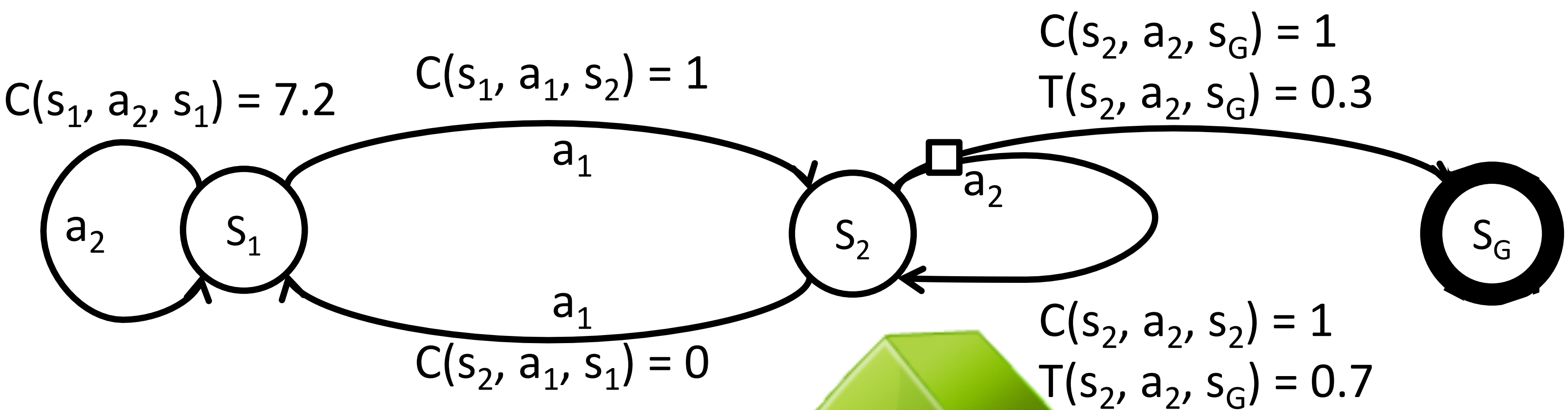
No dead ends allowed!

SSP MDP Example, also not!



❌ *No cost-free "loops" allowed!*

SSP MDP Example



SSP MDPs: Optimality Principle

For an SSP MDP, let:

Exp. Lin. Add. Utility

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

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

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

Then:

– V^* exists and is stationary Markovian, π^* exists and is stationary deterministic Markovian

– For all s:

$$V^*(s) = \min_{a \text{ in } A} [\, \sum_{s' \text{ in } S} T(s, a, s') \, [\, C(s, a, s') + V^*(s') \,] \,]$$

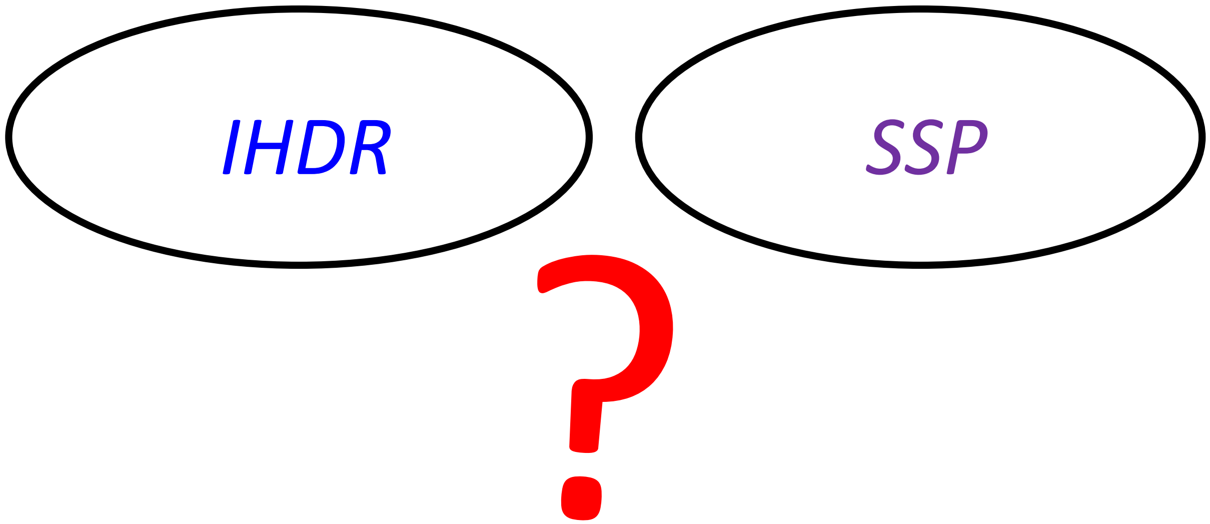
$$\pi^*(s) = \mathrm{argmin}_{a \text{ in } A} [\, \sum_{s' \text{ in } S} T(s, a, s') \, [\, C(s, a, s') + V^*(s') \,] \,]$$

Fundamentals of MDPs

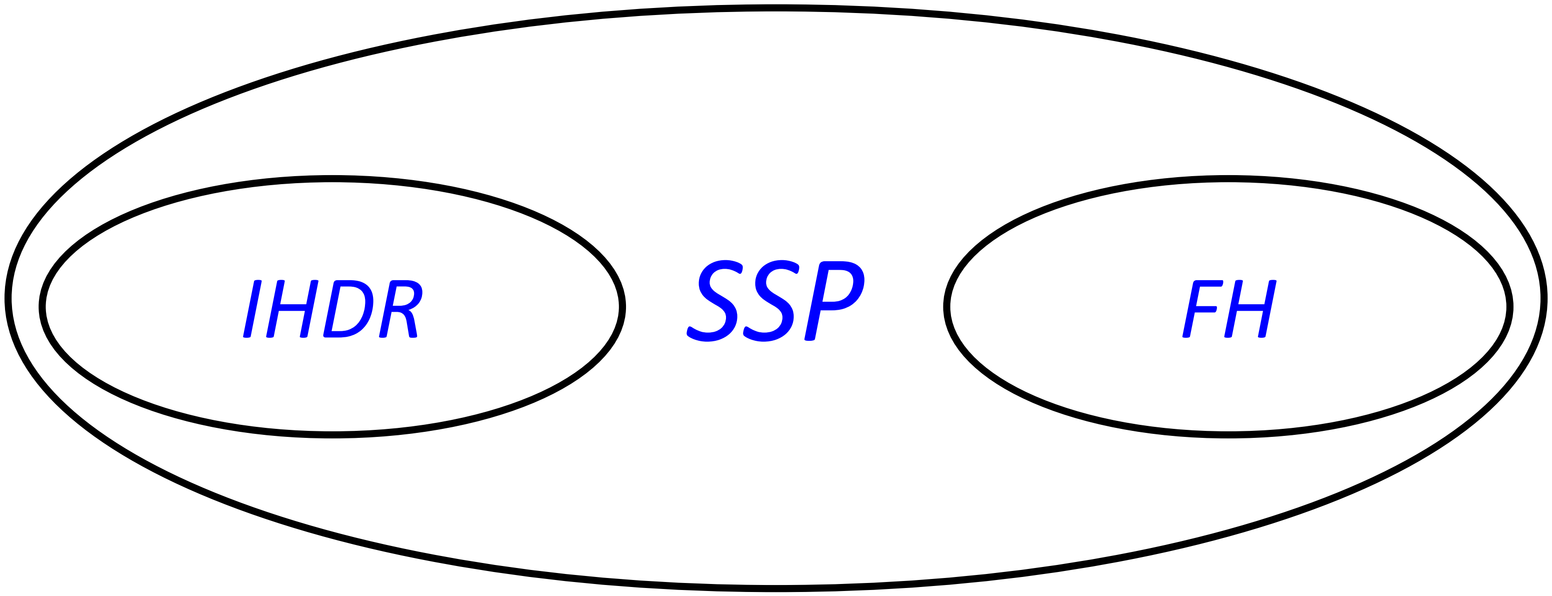
- ✓ General MDP Definition
- ✓ Expected Linear Additive Utility
- ✓ The Optimality Principle
- ✓ Finite-Horizon MDPs
- ✓ Infinite-Horizon Discounted-Reward MDPs
- ✓ Stochastic Shortest-Path MDPs
- **A Hierarchy of MDP Classes**
- Factored MDPs
- Computational Complexity

SSP and Other MDP Classes

E.g., Indefinite-horizon discounted reward

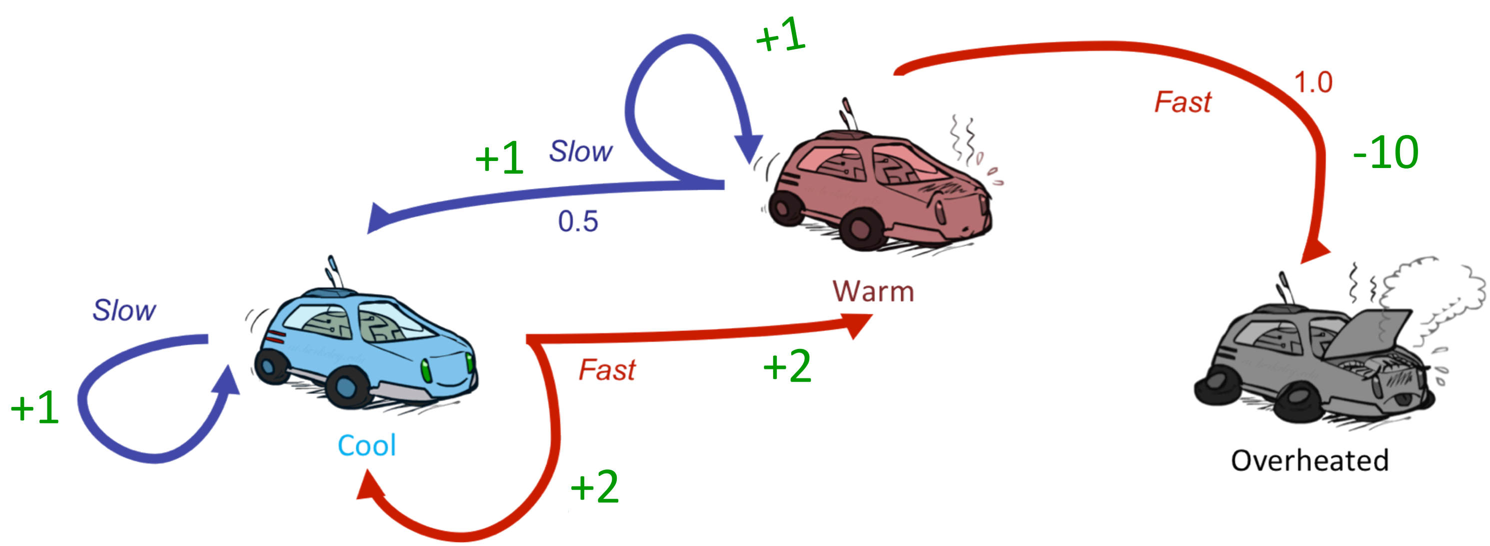


SSP and Other MDP Classes



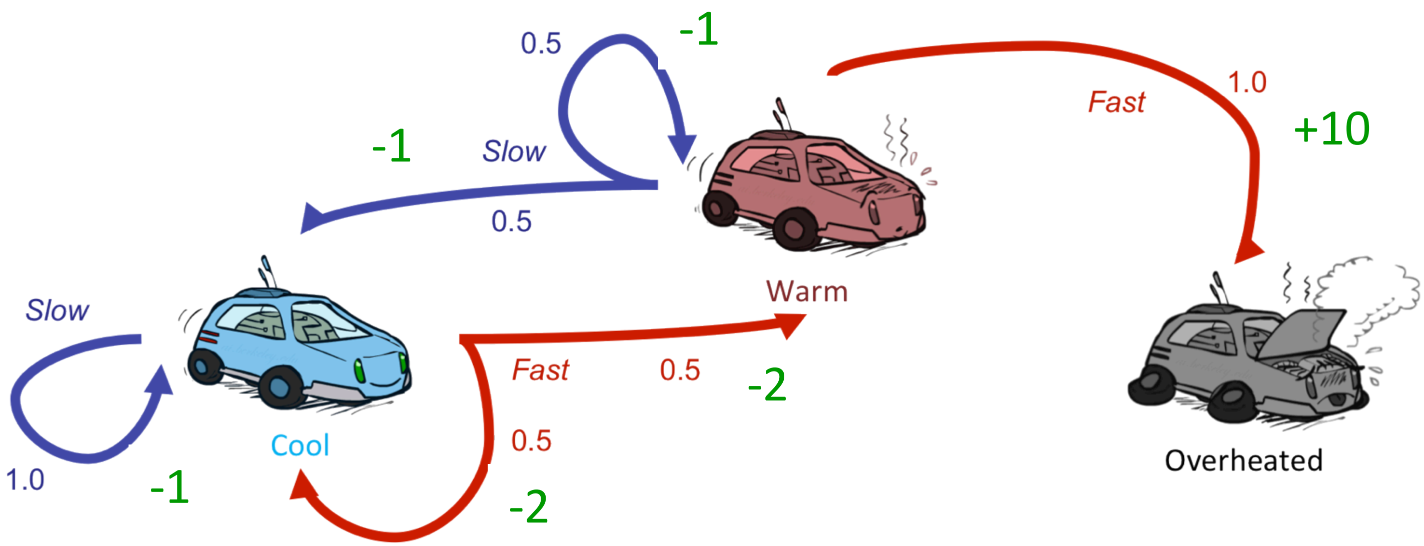
- ***FH* => *SSP***: turn all states (s, L) into goals
- ***IHDR* => *SSP***: add γ-probability transitions to goal
- **Will concentrate on *SSP* in the rest of the tutorial**

IHDR → SSP



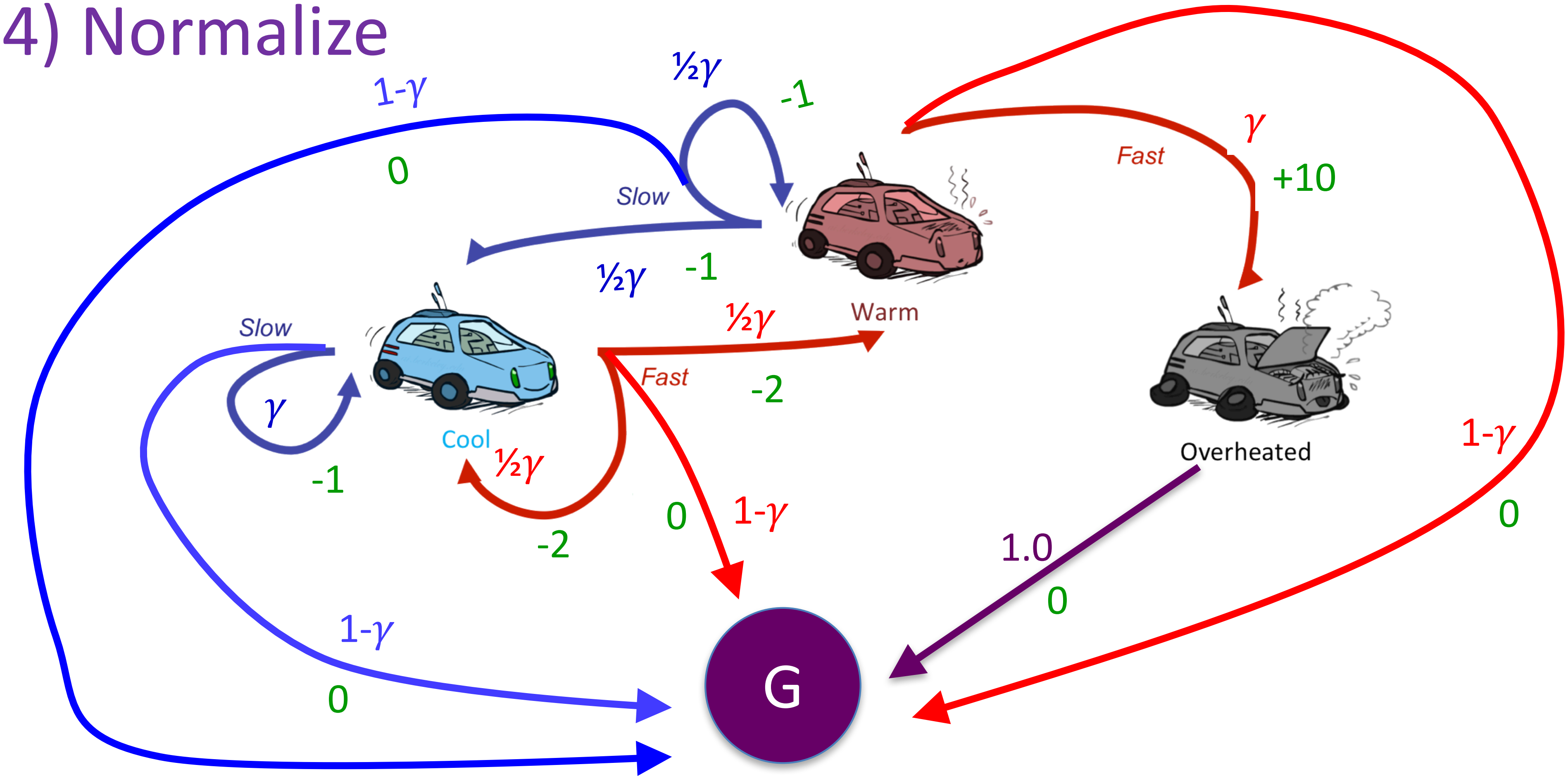
IHDR → SSP

1) Invert rewards to costs



IHDR → SSP

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



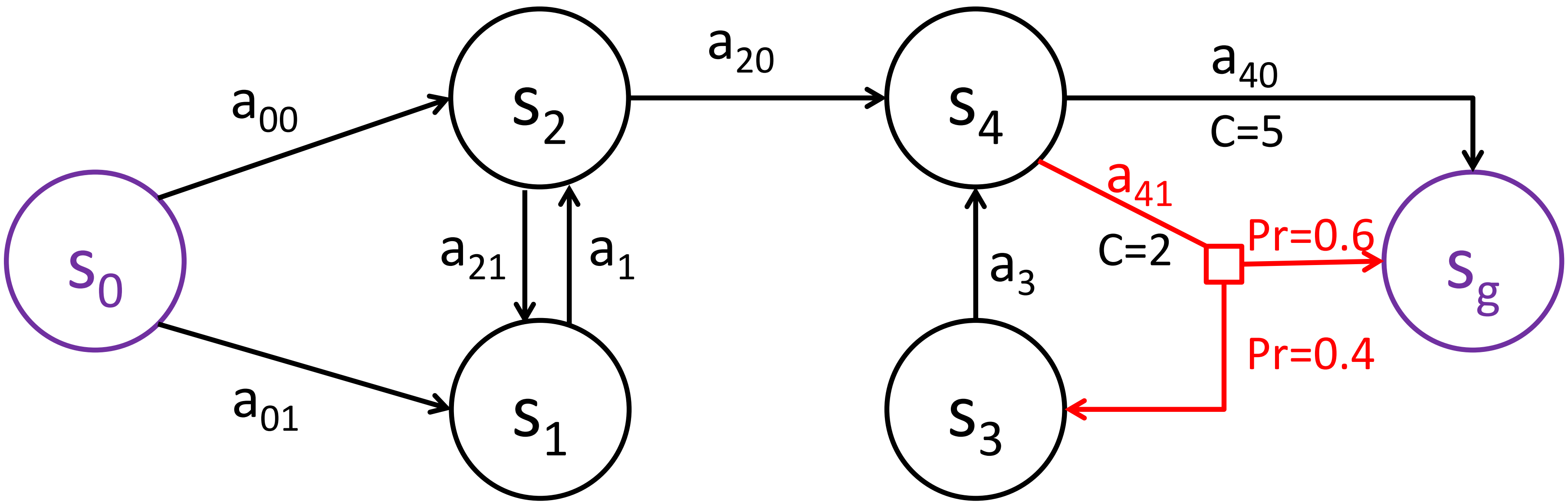
Computational Complexity of MDPs

- **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!

Computational Complexity of MDPs

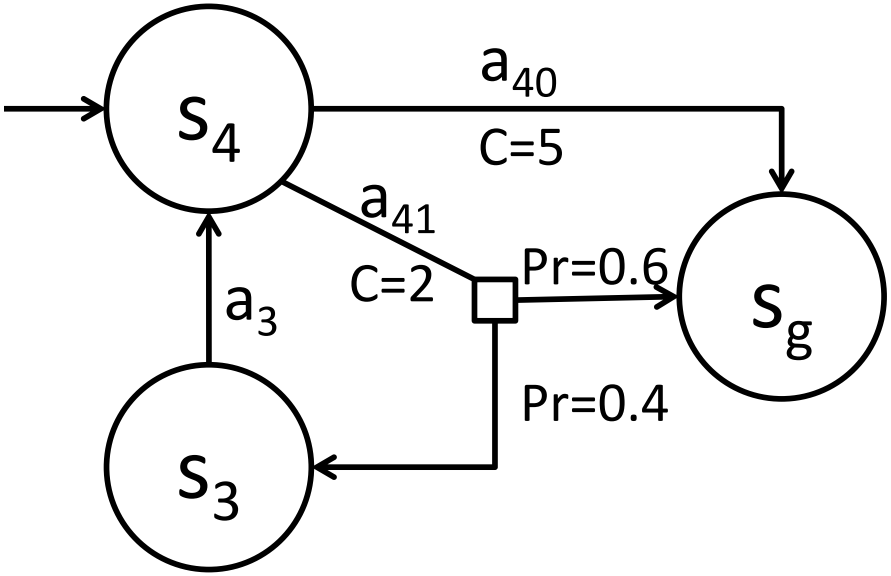
- **Bad news**:
  - Solving *FH, IHDR, SSP* in factored representation is *EXPTIME*-complete!
  - Flat representation doesn't make MDPs harder to solve, it makes big ones easier to describe.

Running Example

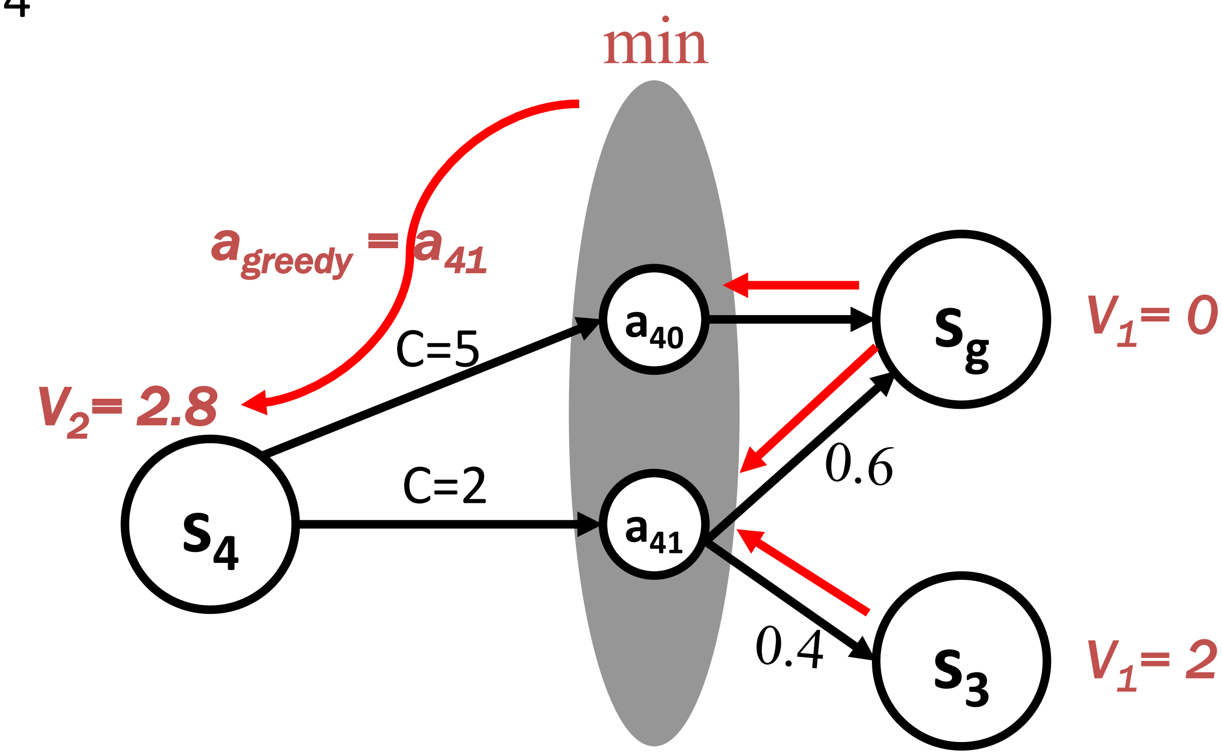


All costs 1 unless otherwise marked

Bellman Backup



$Q_2(s_4,a_{40}) = 5 + 0$
$Q_2(s_4,a_{41}) = 2 + 0.6\times 0$
$+ 0.4\times 2$
$= 2.8$



Value Iteration [Bellman 57]

No restriction on initial value function

initialize V_0 arbitrarily for each state

$n \leftarrow 0$

repeat

  $n \leftarrow n + 1$

  foreach $s \in \mathcal{S}$ **do**

    compute $V_n(s)$ using Bellman backup at s

    compute $\text{residual}_n(s) = |V_n(s) - V_{n-1}(s)|$

  end

until $\max_{s \in \mathcal{S}} \text{residual}_n(s) < \epsilon$;

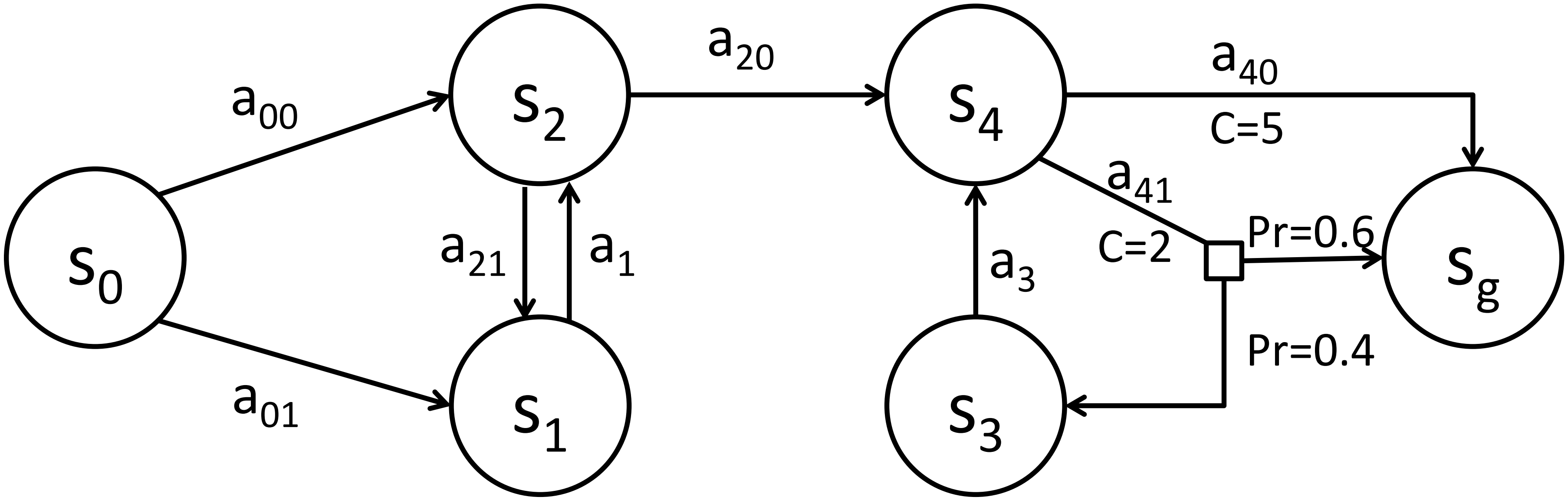
return greedy policy: $\pi^{V_n}(s) = \text{argmin}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') \left[\mathcal{C}(s, a, s') + V_n(s')\right]$

iteration n

ε-consistency

termination condition

Running Example



n	$V_n(s_0)$	$V_n(s_1)$	$V_n(s_2)$	$V_n(s_3)$	$V_n(s_4)$
0	3	3	2	2	1
1	3	3	2	2	2.8
2	3	3	3.8	3.8	2.8
3	4	4.8	3.8	3.8	3.52
4	4.8	4.8	4.52	4.52	3.52
5	5.52	5.52	4.52	4.52	3.808
20	5.99921	5.99921	4.99969	4.99969	3.99969

Convergence & Optimality

- For an SSP MDP, $\forall\ s \in S$,

  $\lim_{n \to \infty} V_n(s) = V^*(s)$

  irrespective of the initialization.

VI → Asynchronous VI

- Is backing up ***all*** states in an iteration essential?
  - No!
- States may be backed up
  - as many times
  - in any order
- If no state gets starved
  - convergence properties still hold!!

Residual *wrt* Value Function V (Res^V)

- Residual at *s* with respect to *V*
  - magnitude($\Delta V(s)$) after one Bellman backup at s

$$Res^V(s) = \left| V_i(s) - \min_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} T(s,a,s')[C(s,a,s') + V_i(s')] \right|$$

- Residual *wrt* respect to *V*
  - max residual
  - $Res^V = \max_s (Res^V(s))$

$Res^V < \epsilon$
(ϵ-consistency)

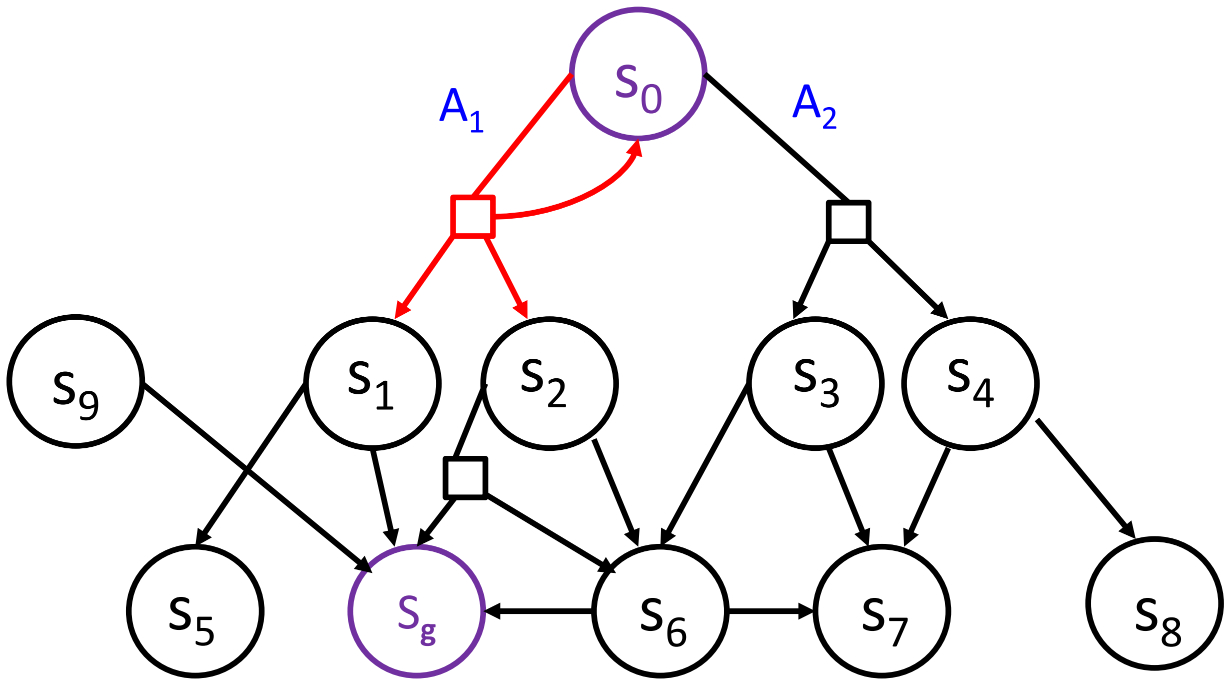
(General) Asynchronous VI

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

Heuristic Search Algorithms

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

Notation



Heuristic Search

- Insight 1
  - knowledge of a start state 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*)

Model

- SSP (as before) with an additional start state s_0
  - denoted by SSP_{s0}

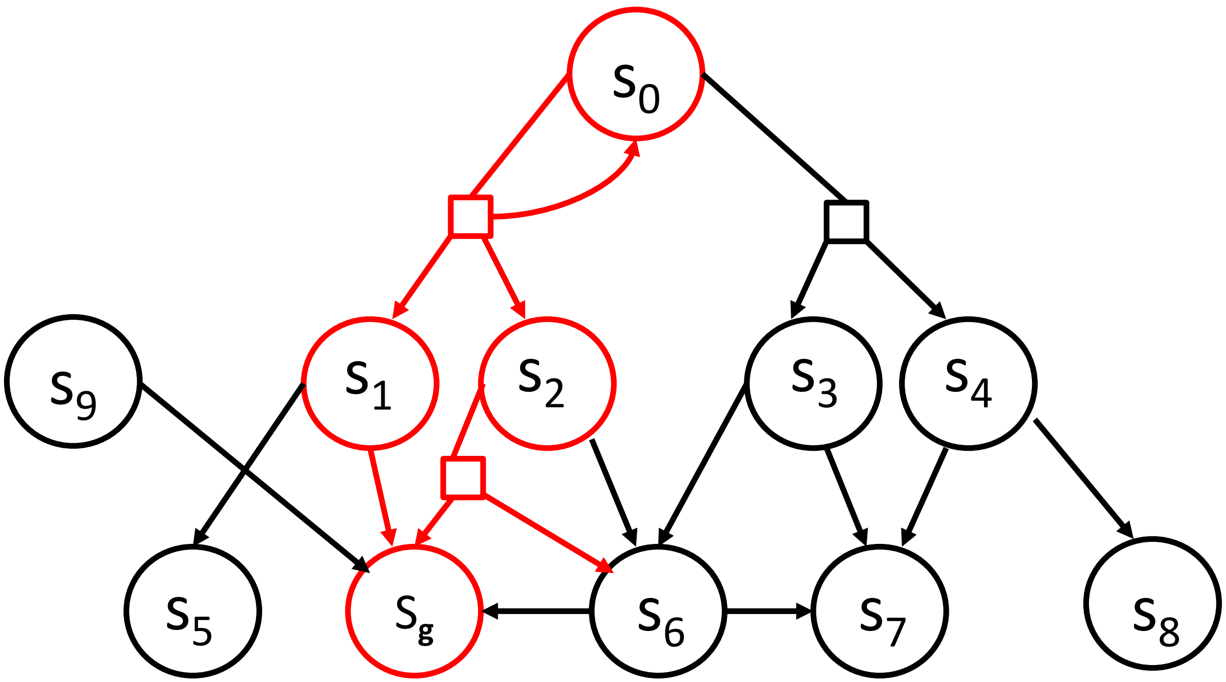
- What is the solution to an SSP_{s0}
- Policy ($S \rightarrow A$)?
  - are states that are not reachable from s_0 relevant?
  - states that are never visited (even though reachable)?

Partial Policy

- Define *Partial policy*
  - π: S' → *A*, where S' $\subseteq$ *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*

Partial policy closed wrt s_0

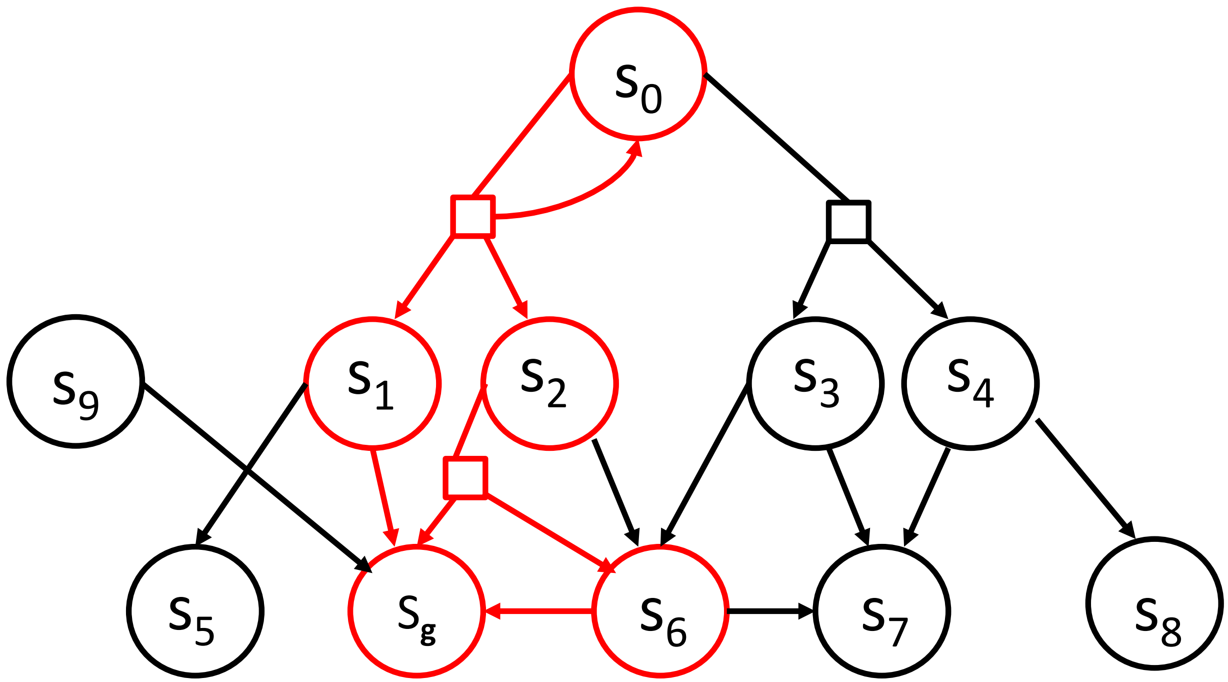


a_1 is left action
a_2 is on right

Is this policy closed wrt s_0?

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

Partial policy closed wrt s_0

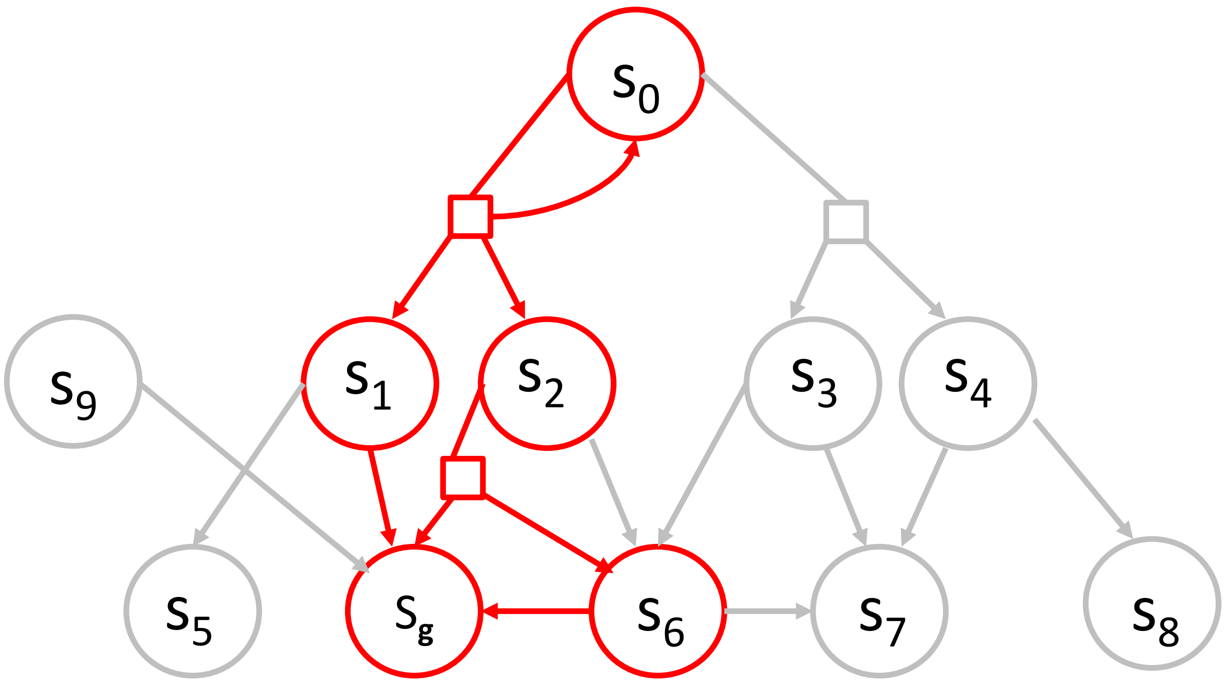


a_1 is left action
a_2 is on right

Is this policy closed wrt s_0?

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

Policy Graph of π_{s0}



a_1 is left action

a_2 is on right

$\pi_{s0}(s_0) = a_1$

$\pi_{s0}(s_1) = a_2$

$\pi_{s0}(s_2) = a_1$

$\pi_{s0}(s_6) = a_1$

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 $\pi^V_{s_0}$
- Define *greedy policy graph*
  - Policy graph of $\pi^V_{s_0}$: denoted by $G^V_{s_0}$

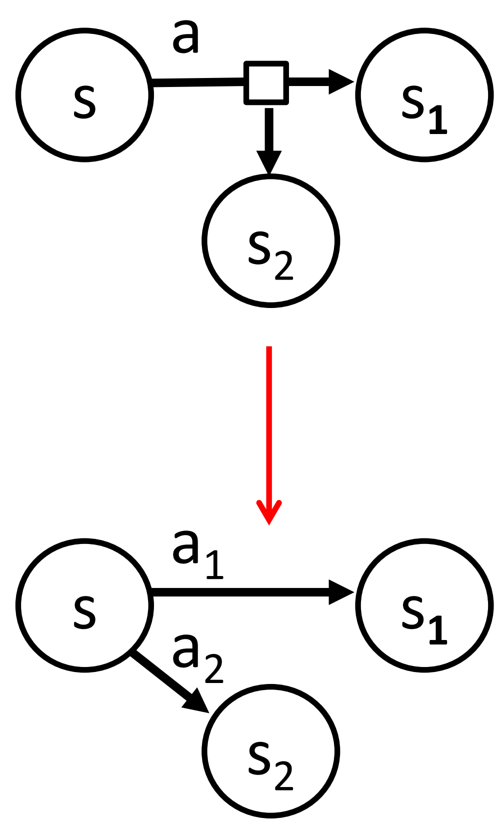
Heuristic Function

- $h(s): S \rightarrow \mathbb{R}$
  - 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)
  - $h(s) \leq V^*(s)$

Admissible Heuristics

- Basic idea
  - Relax probabilistic domain to deterministic domain
  - Use heuristics(classical planning)



- All-outcome Determinization
  - For each outcome create a different action
- Admissible Heuristics
  - Cheapest cost solution for determinized domain
  - Classical heuristics over determinized domain

Heuristic Search Algorithms

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

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 $V(s)$ (perform Bellman backups)
end
return a π^V

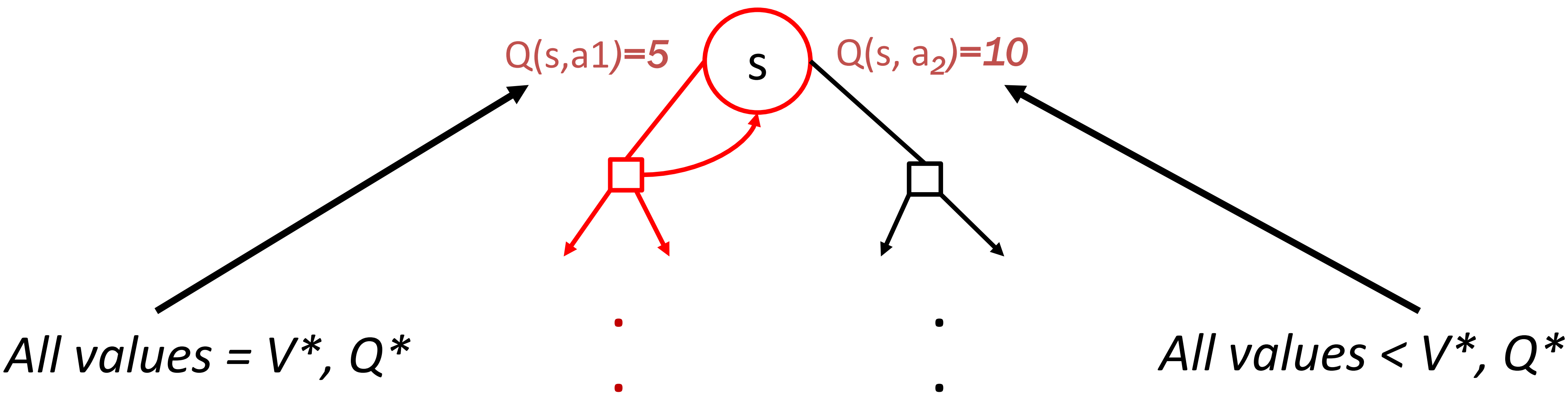
- Convergence to V* is guaranteed
  - if heuristic function is admissible
  - ~no state gets starved in ∞ FIND steps

F&R and Monotonicity

- $V_k \leq_p V^* \Rightarrow V_n \leq_p V^*$ (V_n monotonic from below)

  If h is admissible: $V_0 = h(s) \leq_p V^*$

  $\Rightarrow V_n \leq_p V^*$ ($\forall$ n)



$Q^*(s,a_1) < Q(s,a_2) < Q^*(s,a_2)$

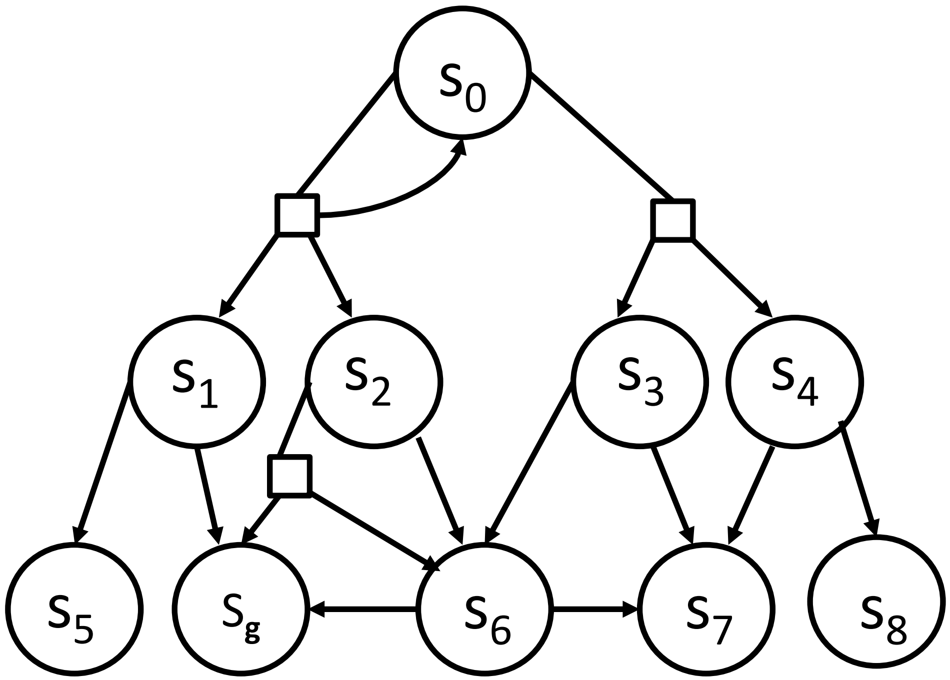
a_2 can't be optimal

Real Time Dynamic Programming

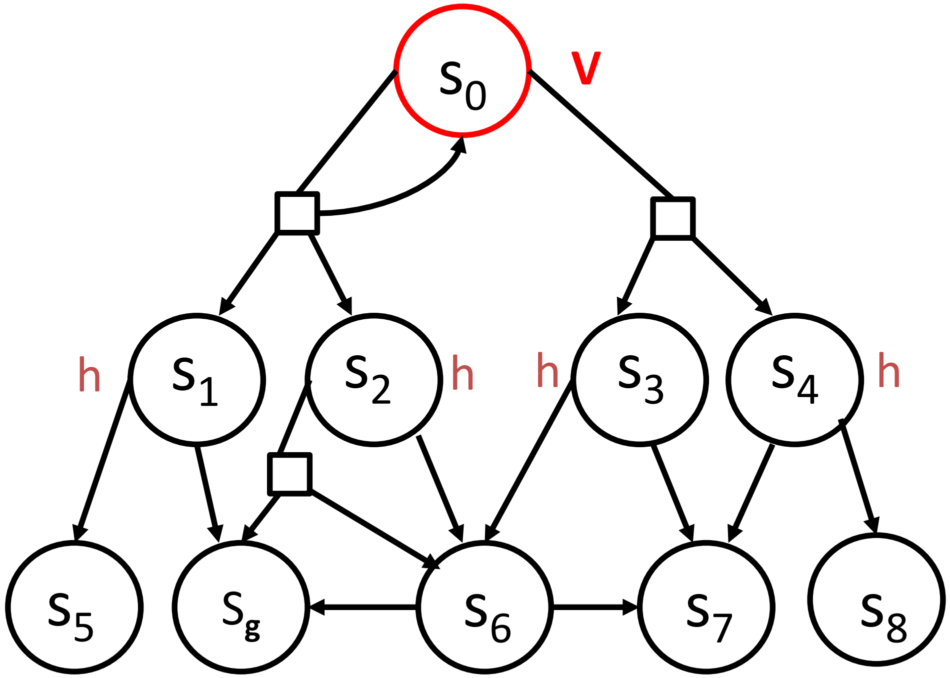
[Barto et al 95]

- 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 → ∞

Trial



Trial



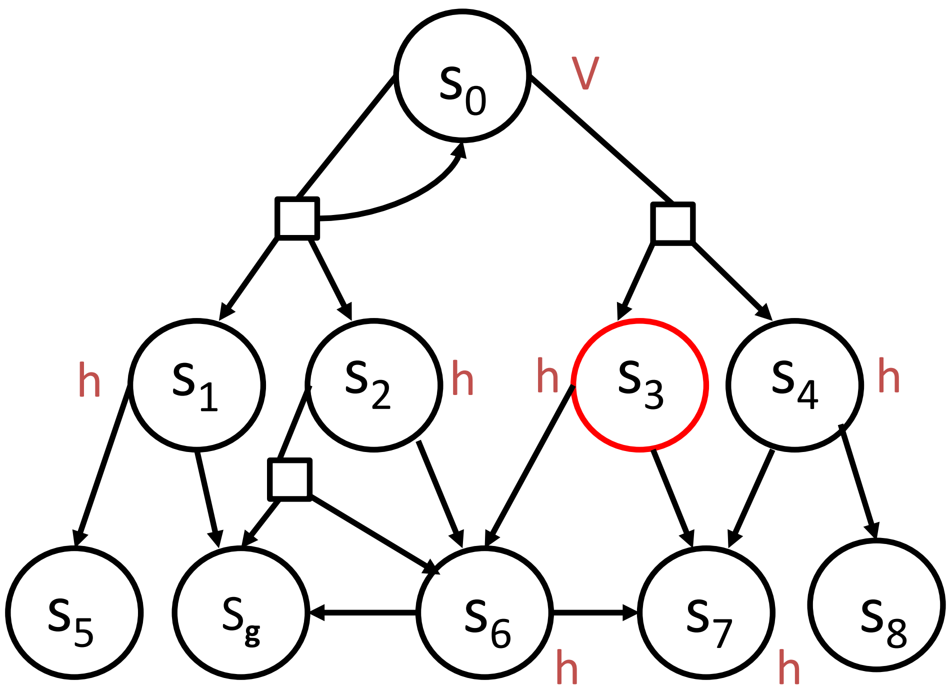
start at start state

repeat

perform a Bellman backup

simulate greedy action

Trial



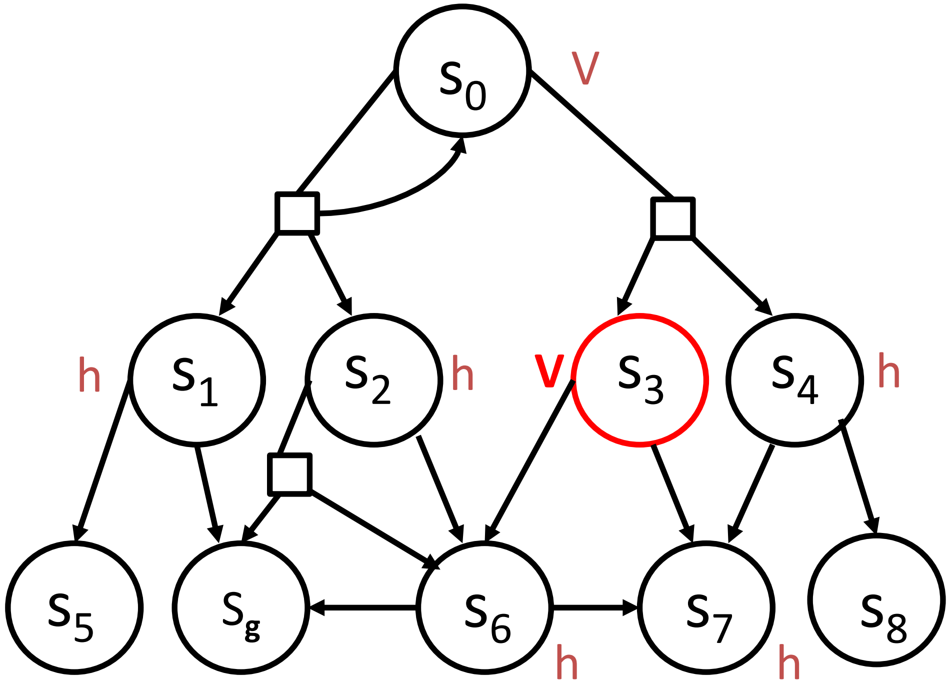
start at start state

repeat

perform a Bellman backup

simulate greedy action

Trial



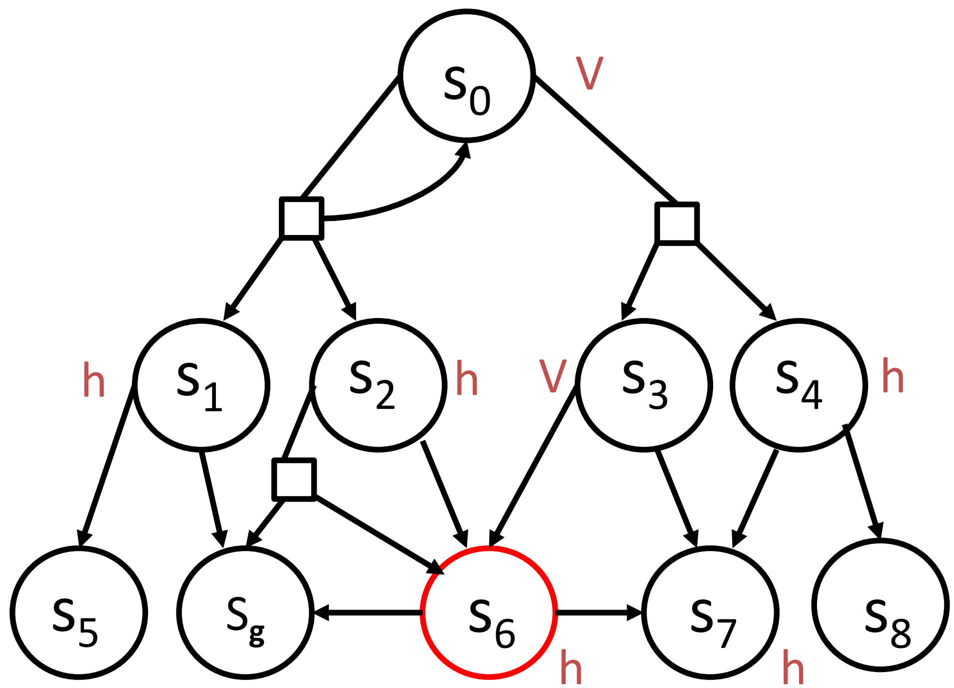
start at start state

repeat

perform a Bellman backup

simulate greedy action

Trial



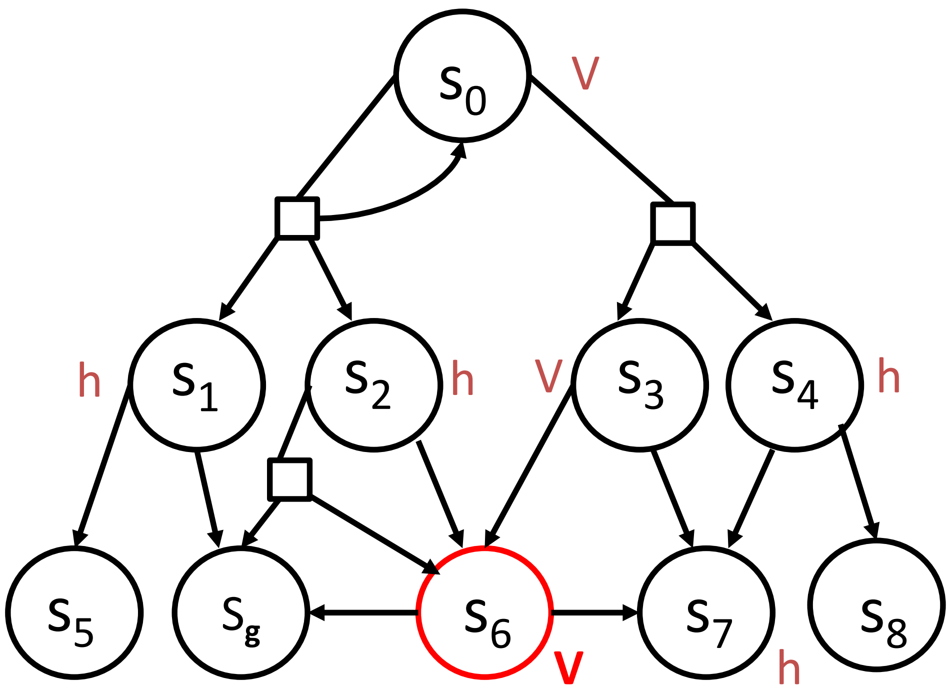
start at start state

repeat

perform a Bellman backup

simulate greedy action

Trial



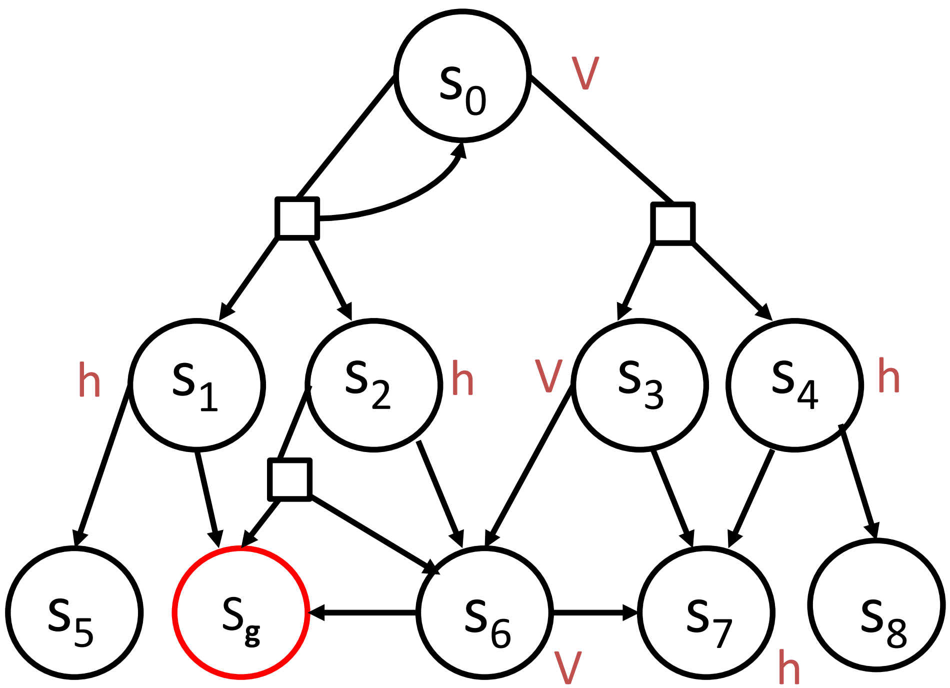
start at start state

repeat

perform a Bellman backup

simulate greedy action

Trial



start at start state

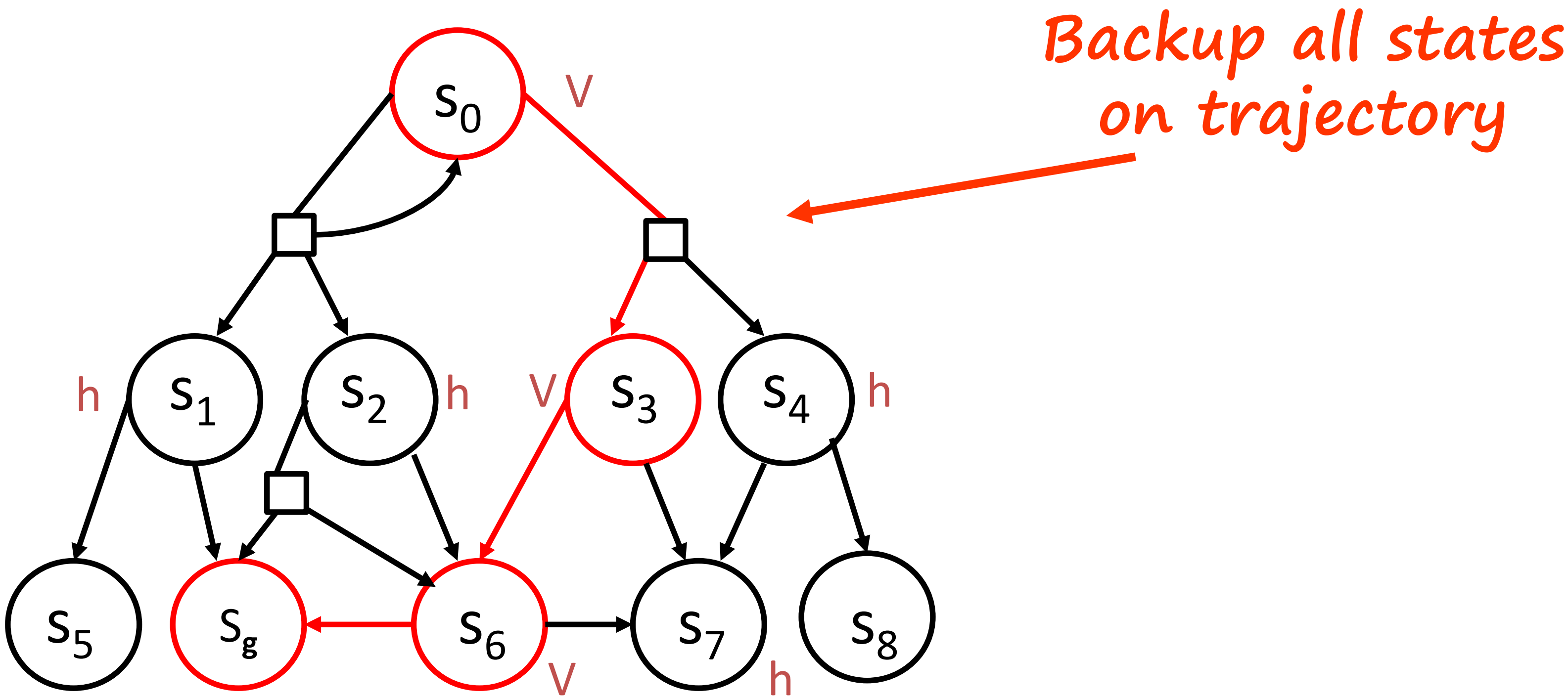
repeat

    perform a Bellman backup

    simulate greedy action

until hit the goal

Trial



RTDP

repeat forever

start at start state

repeat

perform a Bellman backup

simulate greedy action

until hit the goal

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 → ∞

No termination condition!

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

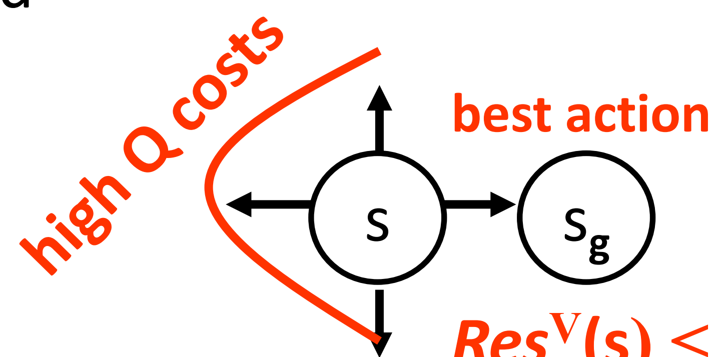
Termination Test Take 1: Labeling

- Admissible heuristic & monotonicity

  $\Rightarrow V(s) \leq V^*(s)$

  $\Rightarrow Q(s,a) \leq Q^*(s,a)$

- Label state, s, as solved
  - if V(s) has converged

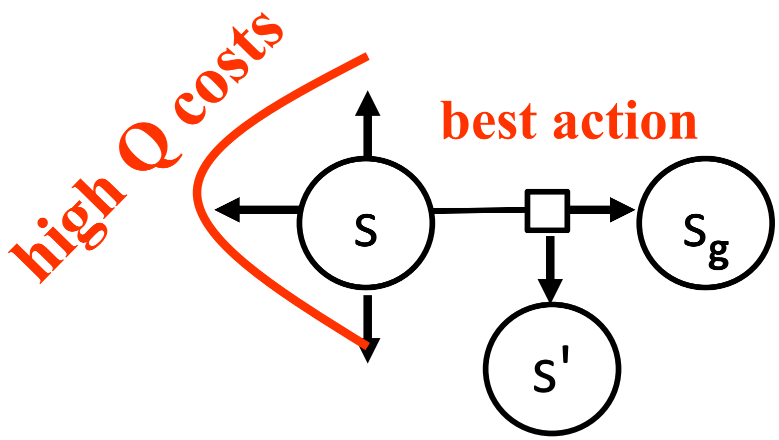


$Res^V(s) < \varepsilon$

$\Rightarrow$ V(s) won't change!

label s as solved

Labeling (contd)



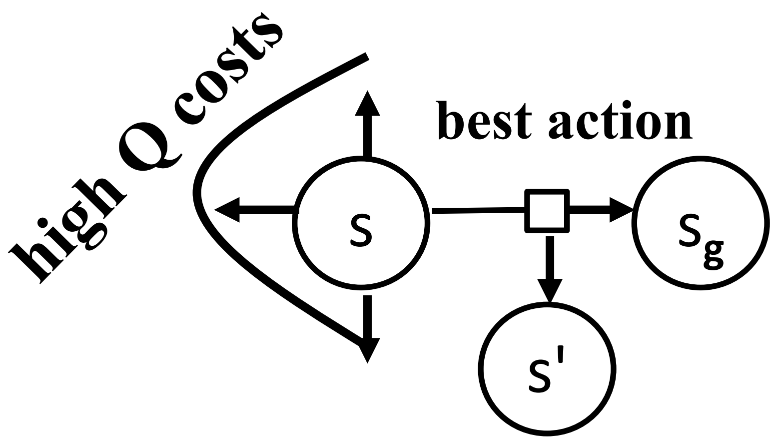
$Res^V(s) < \varepsilon$

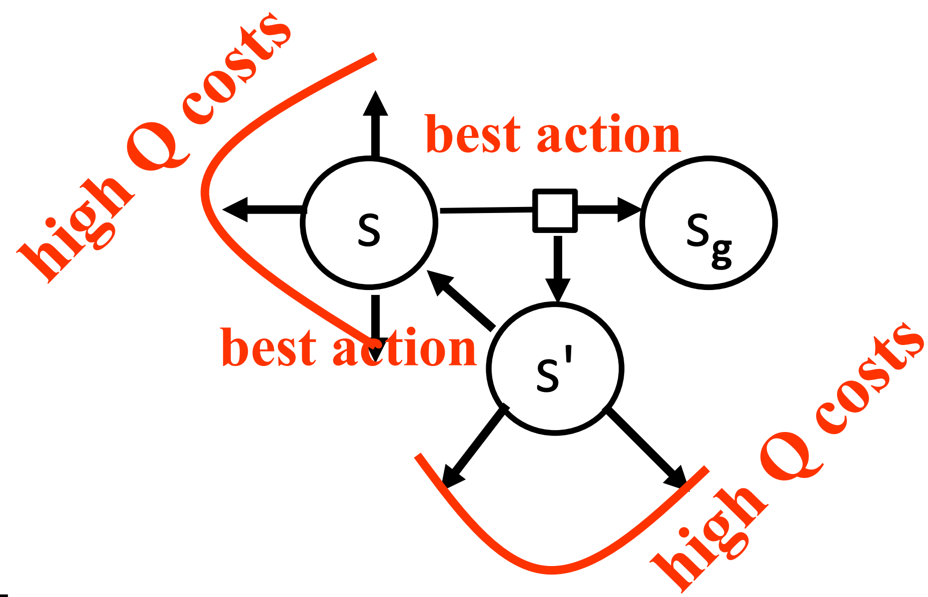
s' already solved

$\Rightarrow$ V(s) won't change!

label s as solved

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

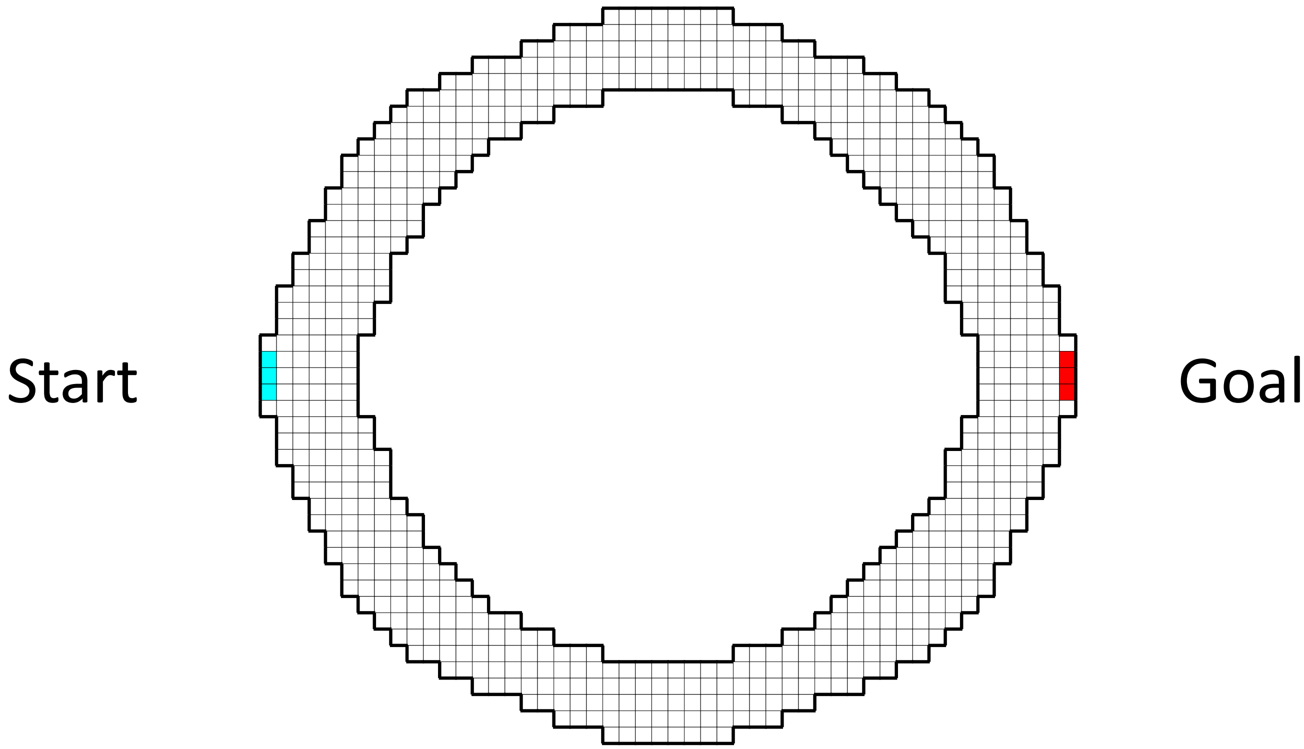
    try to label s as solved

until s_0 is solved

LRTDP

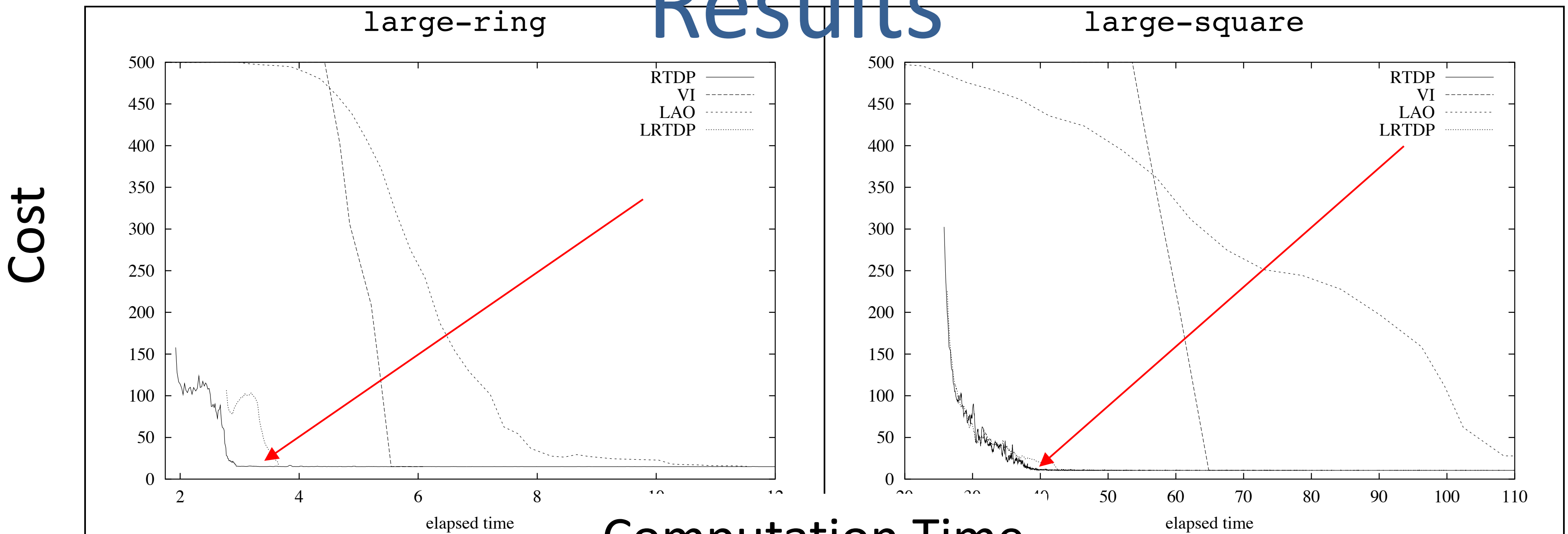
- terminates in finite time
  - due to labeling procedure
- anytime
  - focuses attention on more probable states
- fast convergence
  - focuses attention on unconverged states

LRTDP Experiments



Racetrack Domain

Results



Computation Time

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) is expected to change?
  - sample s’ $\propto$ expected change

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

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^*

Bounded RTDP [McMahan et al 05]

repeat

$\quad$ s $\leftarrow$ s_0

$\quad$ repeat *//trials*

$\quad\quad$ identify a_{greedy} based on V_l

$\quad\quad$ FIND: sample s' $\propto$ T(s, a_{greedy}, s').gap(s')

$\quad\quad$ s $\leftarrow$ s'

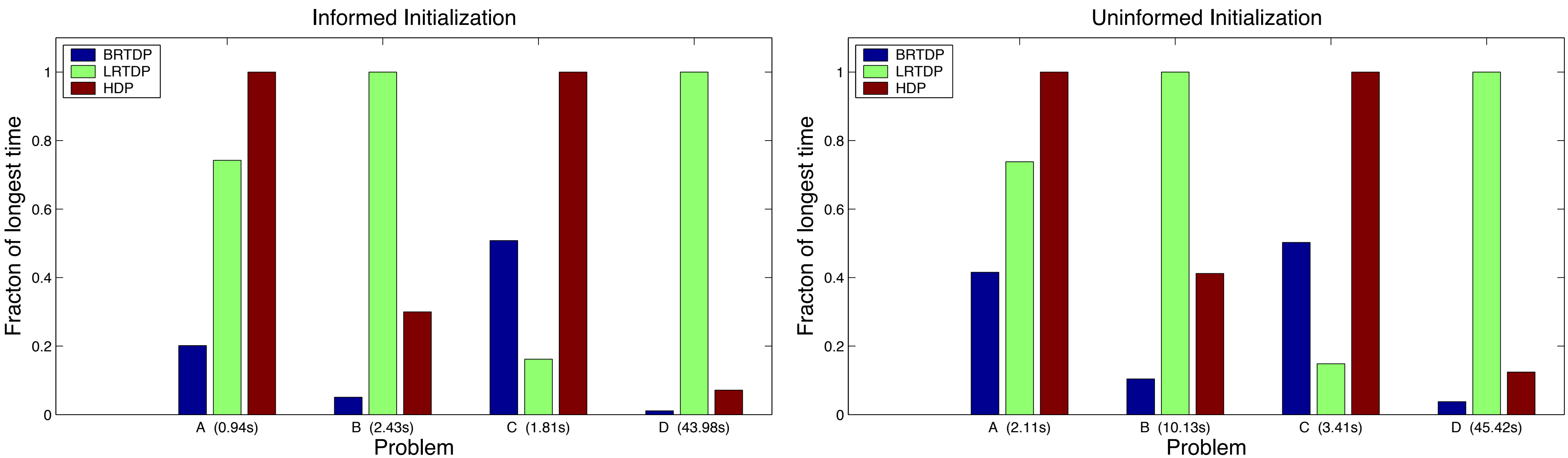
$\quad$ until gap(s) < ε

$\quad$ for all states s in trial in reverse order

$\quad\quad$ REVISE s

until gap(s_0) < ε

BRTDP Results



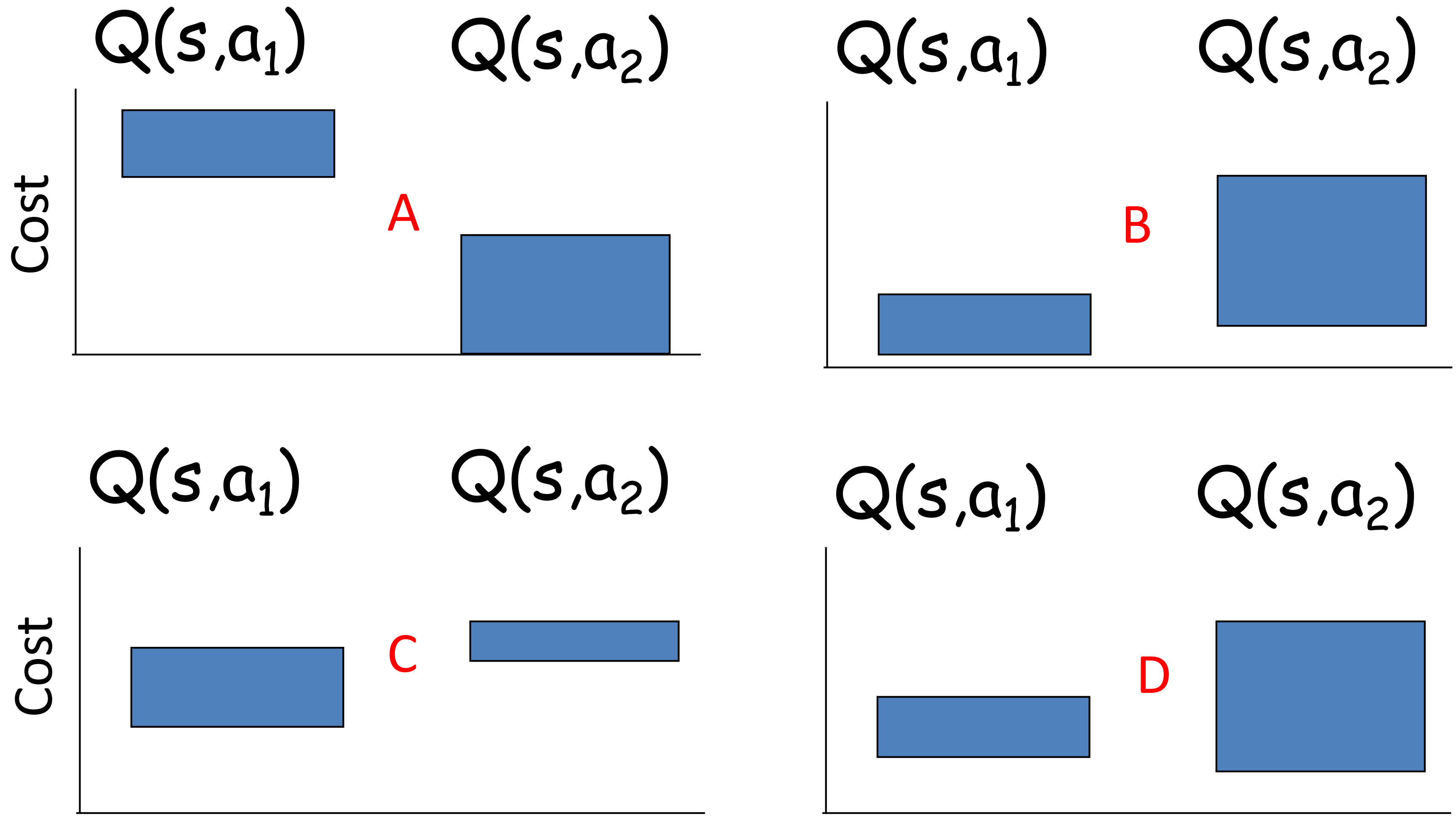
A, B – racetrack; C,D gridworld. A,C have sparse noise; B,D much noise

Is that the best we can do?

Focused RTDP [Smith&Simmons 06]

- Similar to Bounded RTDP except
  - a more sophisticated definition of priority that combines gap and prob. of reaching the state
  - adaptively increasing the max-trial length

Picking a Successor Take 3



[Slide adapted from Scott Sanner]

Value of Perfect Information RTDP [Sanner et al 09]

- What is the expected value of knowing V(s')

- Estimates EVPI(s')
  - using Bayesian updates
  - picks s' with maximum EVPI

Focused RTDP Results

Algorithm	large-b	large-b-3	large-b-w	large-ring	large-ring-3	large-ring-w
RTDP	5.30 (5.19)	10.27 (9.12)	149.07 (190.55)	3.39 (4.81)	8.05 (8.56)	16.44 (91.67)
LRTDP	1.21 (3.52)	1.63 (4.08)	1.96 (14.38)	1.74 (5.19)	2.14 (5.71)	3.13 (22.15)
HDP	1.29 (3.43)	1.86 (4.12)	2.87 (15.99)	1.27 (4.35)	2.74 (6.41)	2.92 (20.14)
HDP+L	1.29 (3.75)	1.86 (4.55)	2.87 (16.88)	1.27 (4.70)	2.74 (7.02)	2.92 (21.12)
FRTDP	**0.29 (2.10)**	**0.49 (2.38)**	**0.84 (10.71)**	**0.22 (2.60)**	**0.43 (3.04)**	**0.99 (14.73)**

Figure 1: Millions of backups before convergence with $\epsilon = 10^{-3}$. Each entry gives the number of millions of backups, with the corresponding wallclock time (seconds) in parentheses. The fastest time for each problem is shown in bold.

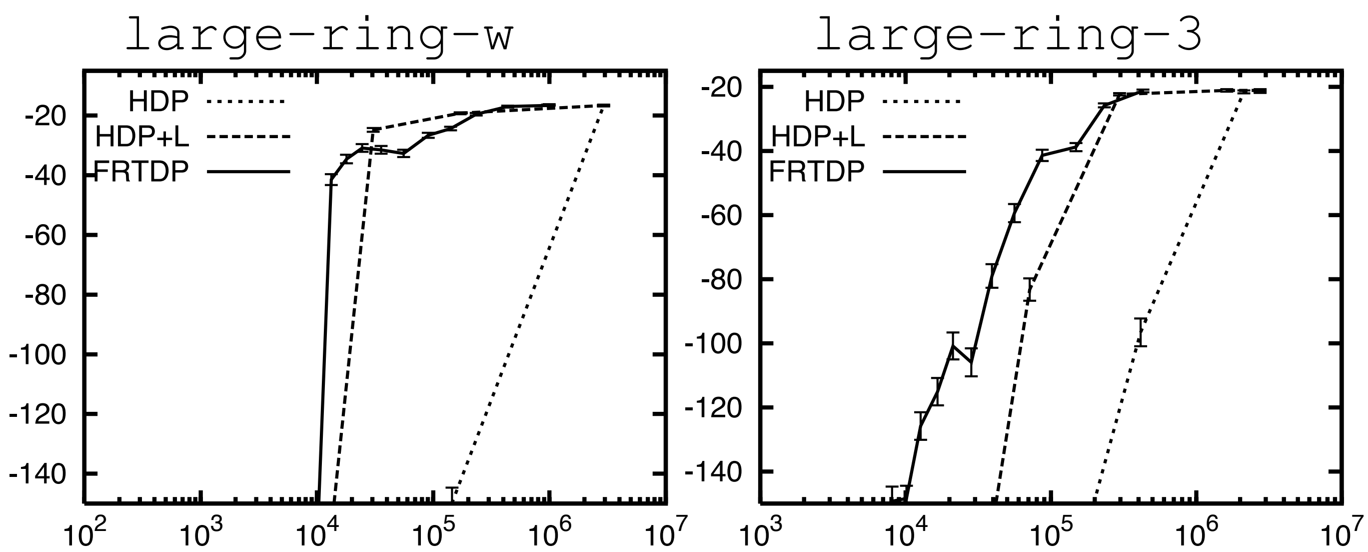


Figure 2: Anytime performance comparison: solution quality vs. number of backups.