

Solving Large POMDPs using Real Time Dynamic Programming

Debabrota Basu

Jaemin Son

**Department of Computer Science,
National University of Singapore**

Reference

Geffner, Hector, and Blai Bonet. "Solving large POMDPs using real time dynamic programming." *In Proc. AAAI Fall Symp. on POMDPs*. 1998.

Outline

- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- MDP
 - RTDP
- POMDP
 - RTDP-BEL
- Experimental Results
- Discussions

Outline

- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- MDP
 - RTDP
- POMDP
 - RTDP-BEL
- Experimental Results
- Discussions

Real world problems

- All information not available *beforehand*
 - Robots should explore to retrieve information
 - Decision has to be made in *real time*
- Not all states need to be considered
 - Subset of states are related to optimal policy from start to goal

Two paradigms- offline vs. online

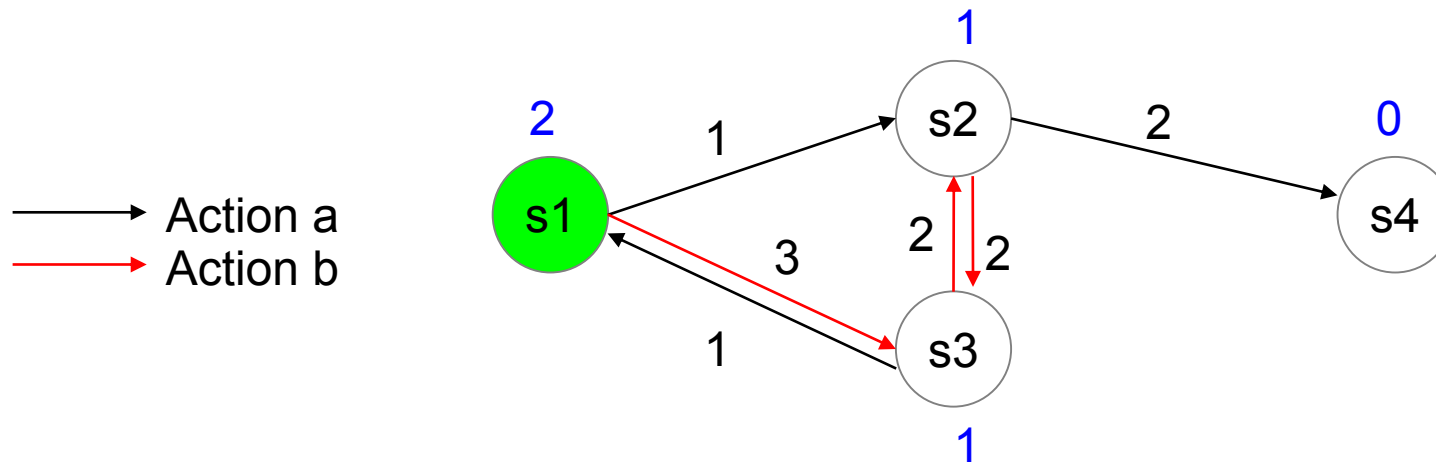
	Offline	Online
Environment	Static	Dynamic
Search space	Complete state space	Subset of state space
Optimality	Globally optimal	Not globally optimal
Time constraints	Loose	Strict
Computation Time	Depends on $ S $	Depends on size of lookahead

Outline

- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- MDP
 - RTDP
- POMDP
 - RTDP-BEL
- Experimental Results
- Discussions

Real Time Search

- Greedily choose best action



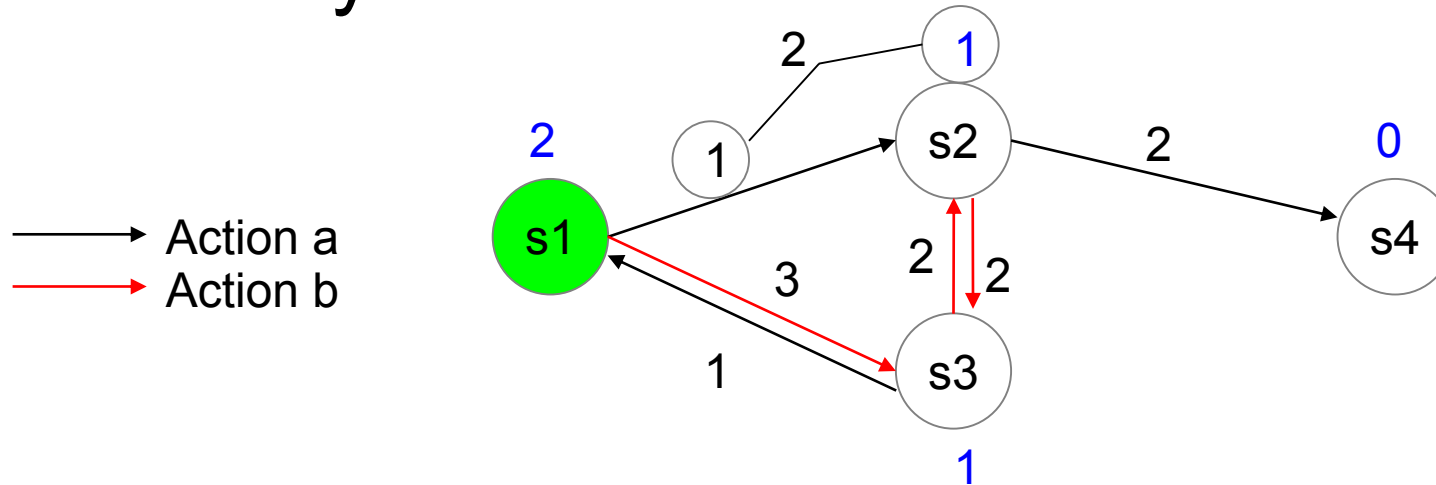
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



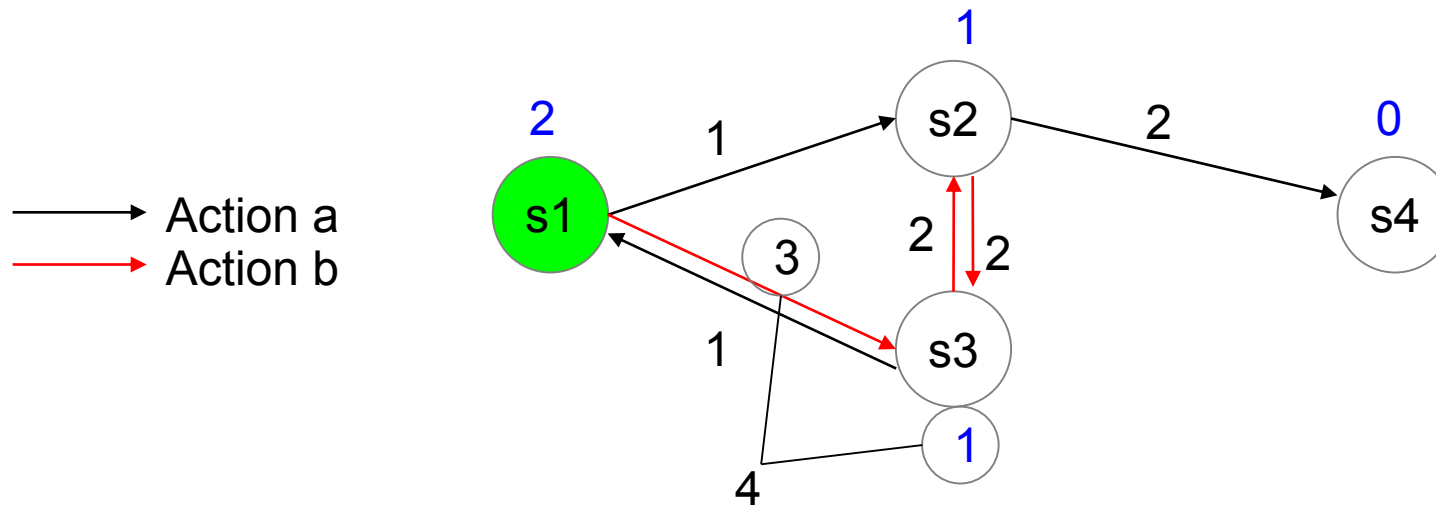
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



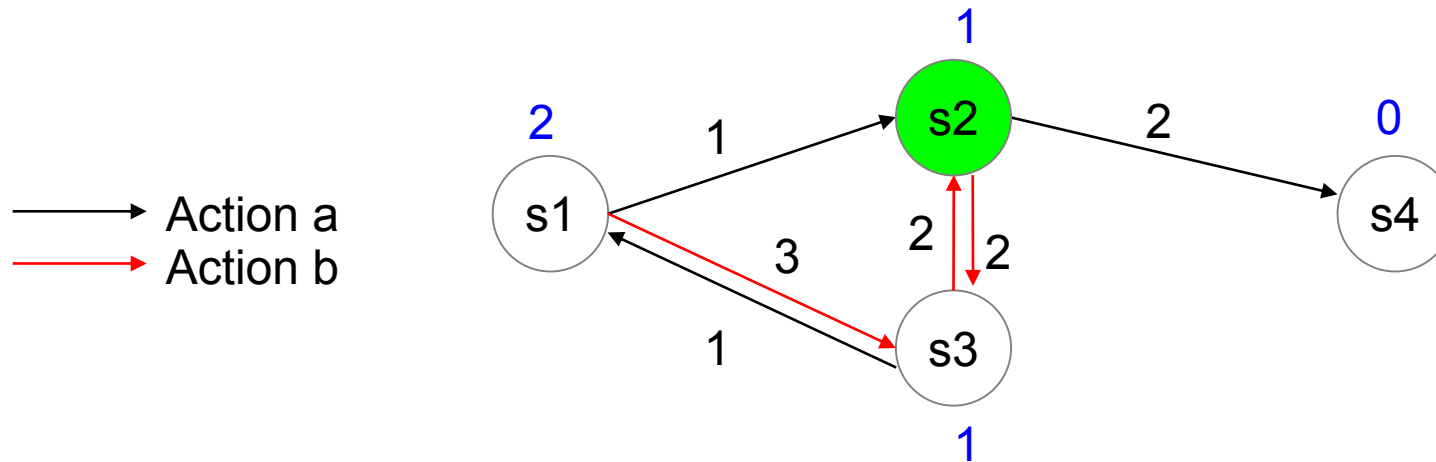
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



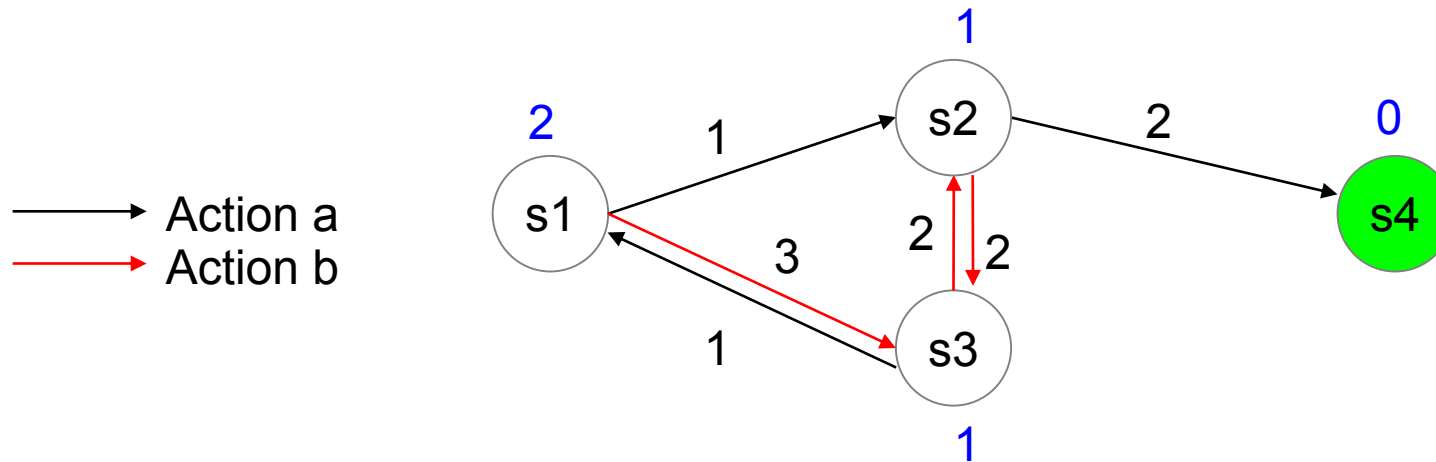
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



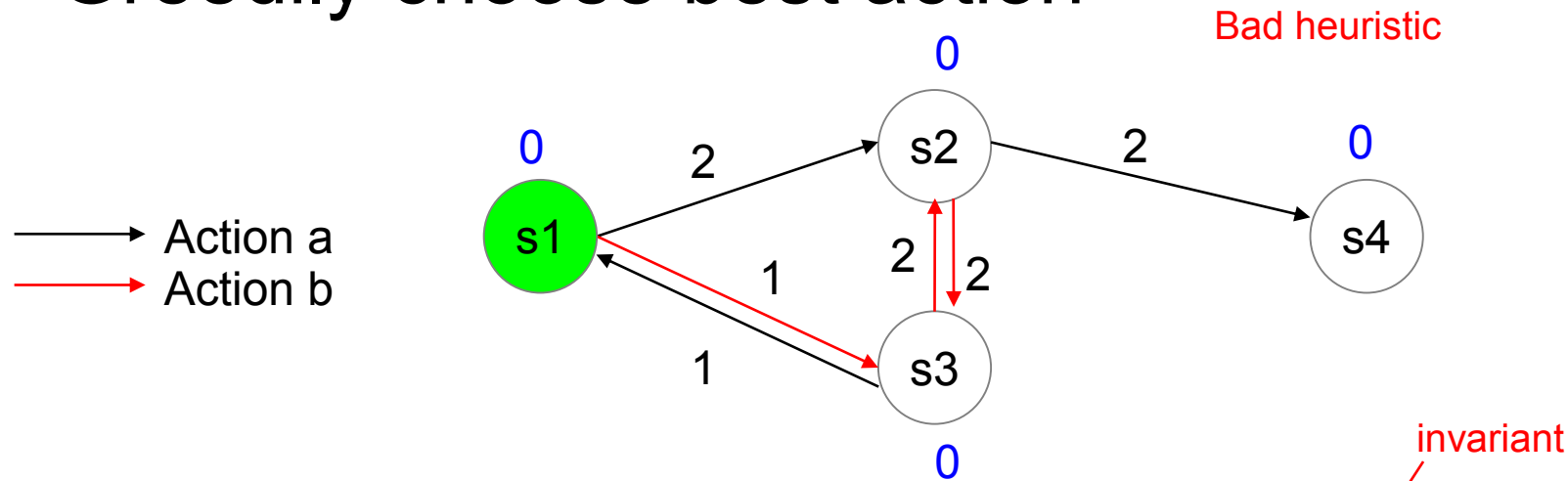
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



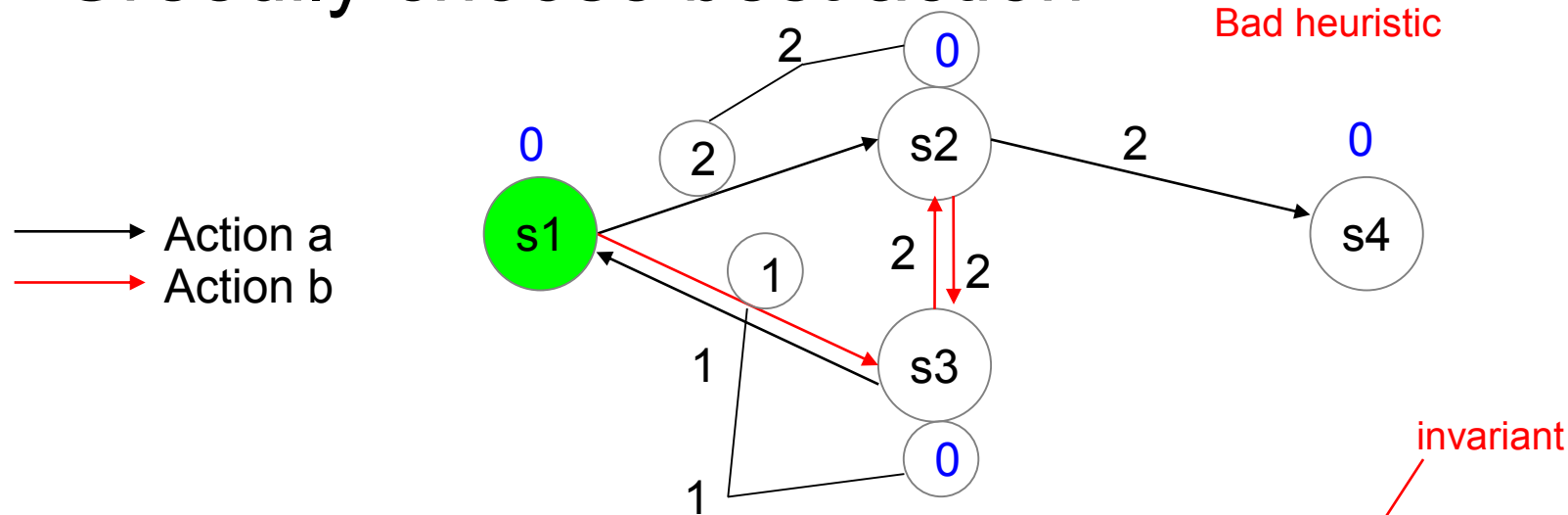
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



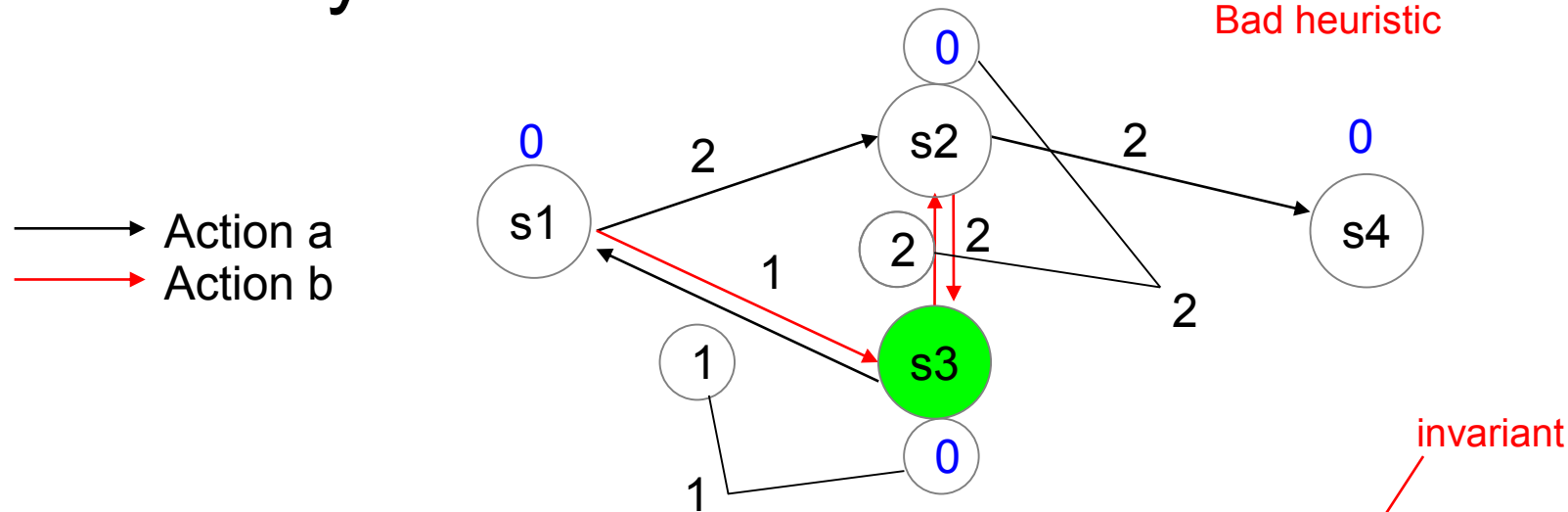
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



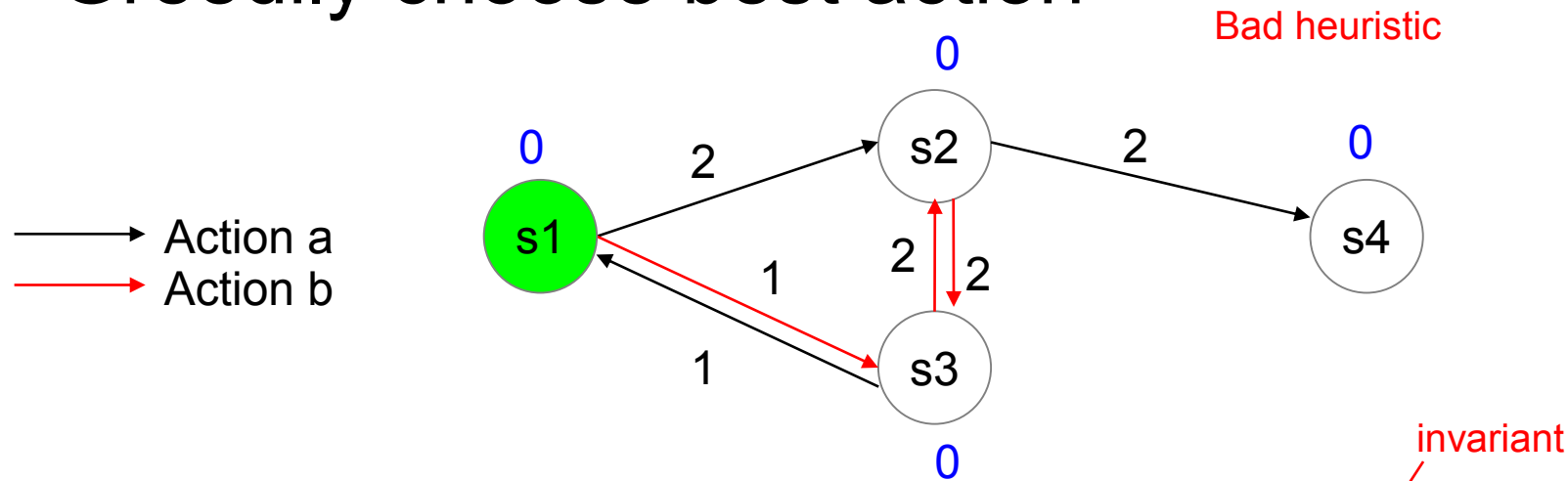
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



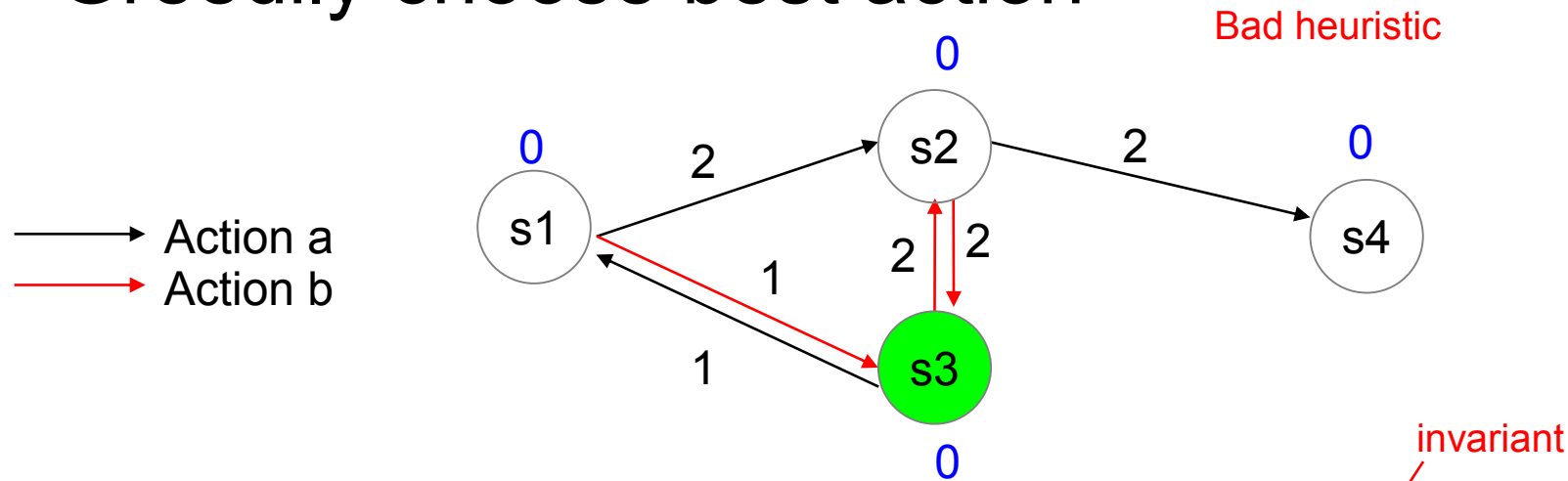
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



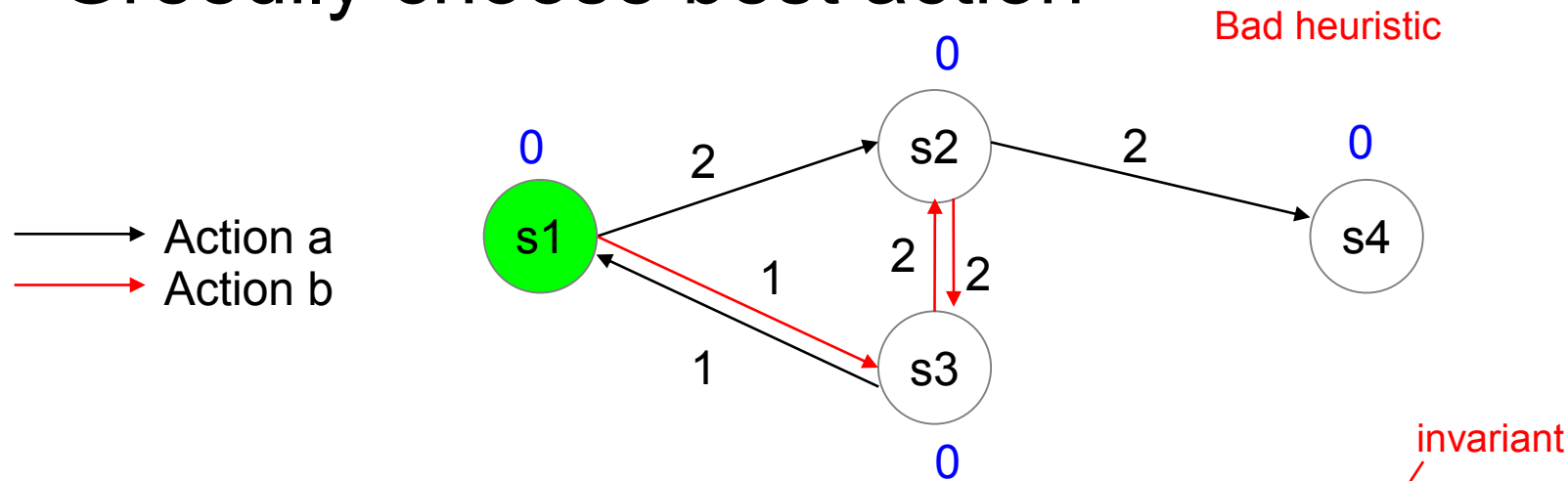
1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Real Time Search

- Greedily choose best action



1. **Evaluate** each action a applicable in current state s as

$$Q(a, s) = \underbrace{c(a, s)}_{\text{Cost (immediate)}} + \underbrace{h(s_a)}_{\text{Heuristic (Future cost)}}$$

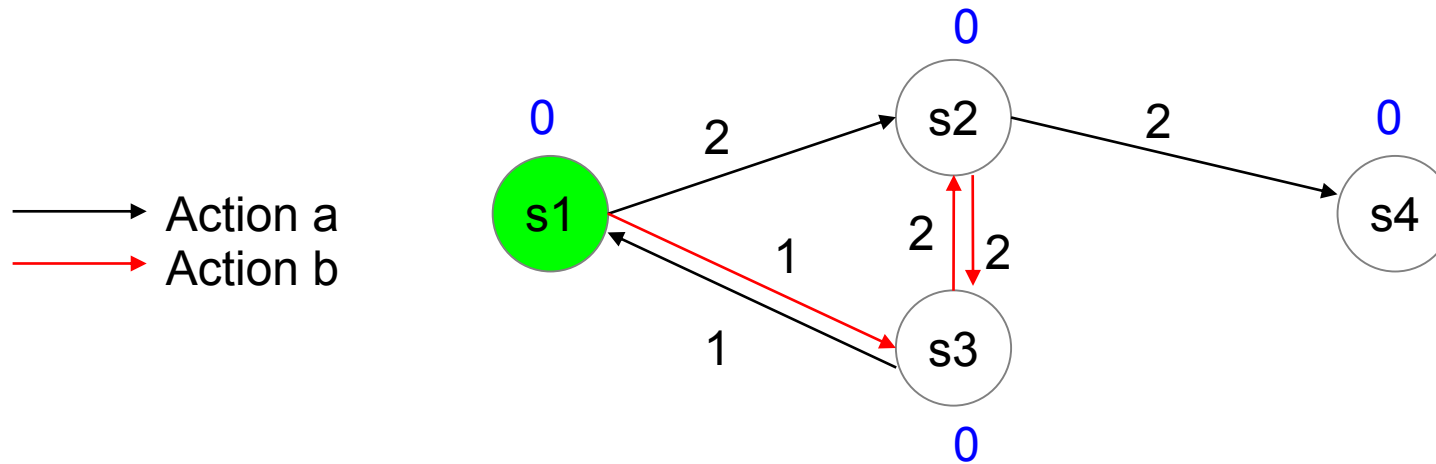
2. **Apply** action a that minimizes $Q(a, s)$, breaking ties randomly
3. **Observe** resulting state s'
4. **Exit** if s' is a goal state, else set s to s' and go to 1

Outline

- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- MDP
 - RTDP
- POMDP
 - RTDP-BEL
- Experimental Results
- Discussions

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly

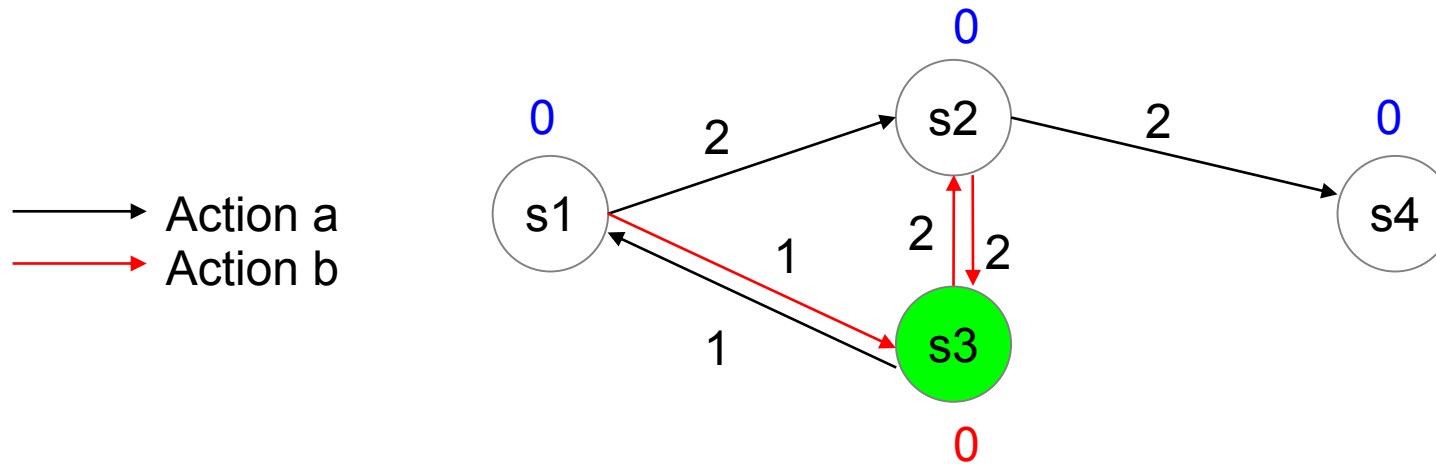
3. Update $V(s)$ to $Q(a, s)$ **LEARNING**

4. Observe resulting state s'

5. Exit: if s' is a goal, else set s to s' and go to 1

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly

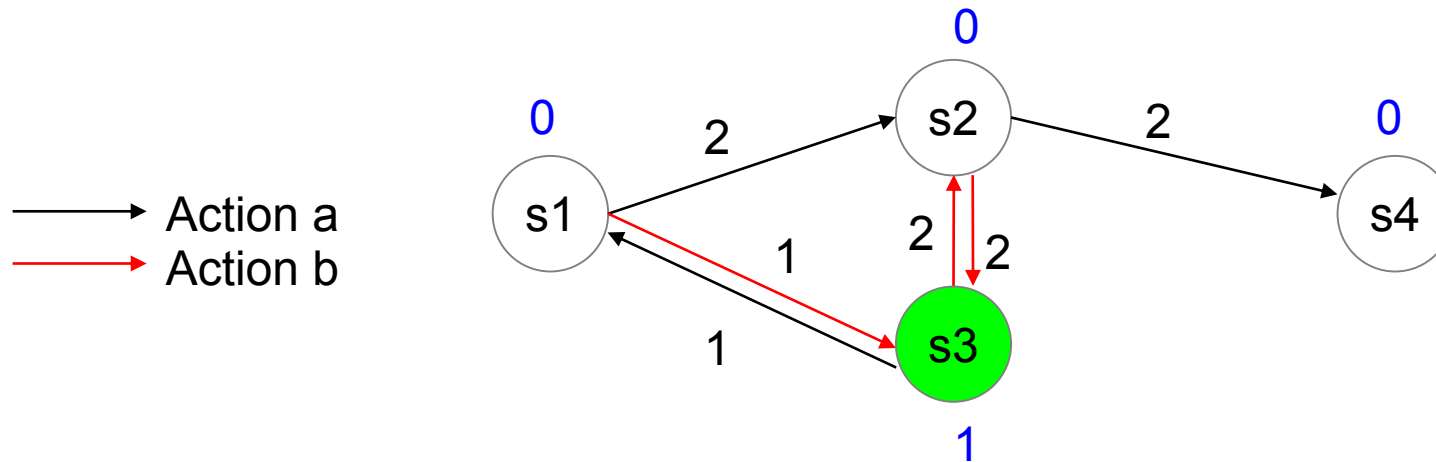
3. Update $V(s)$ to $Q(a, s)$ **LEARNING**

4. Observe resulting state s'

5. Exit: if s' is a goal, else set s to s' and go to 1

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly

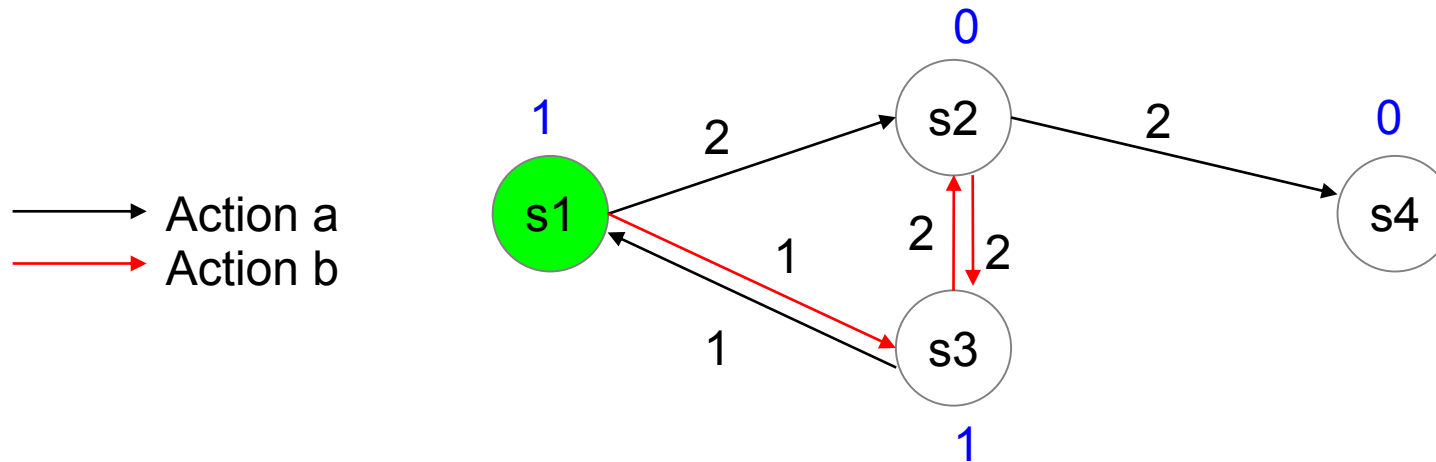
3. Update $V(s)$ to $Q(a, s)$ **LEARNING**

4. Observe resulting state s'

5. Exit: if s' is a goal, else set s to s' and go to 1

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

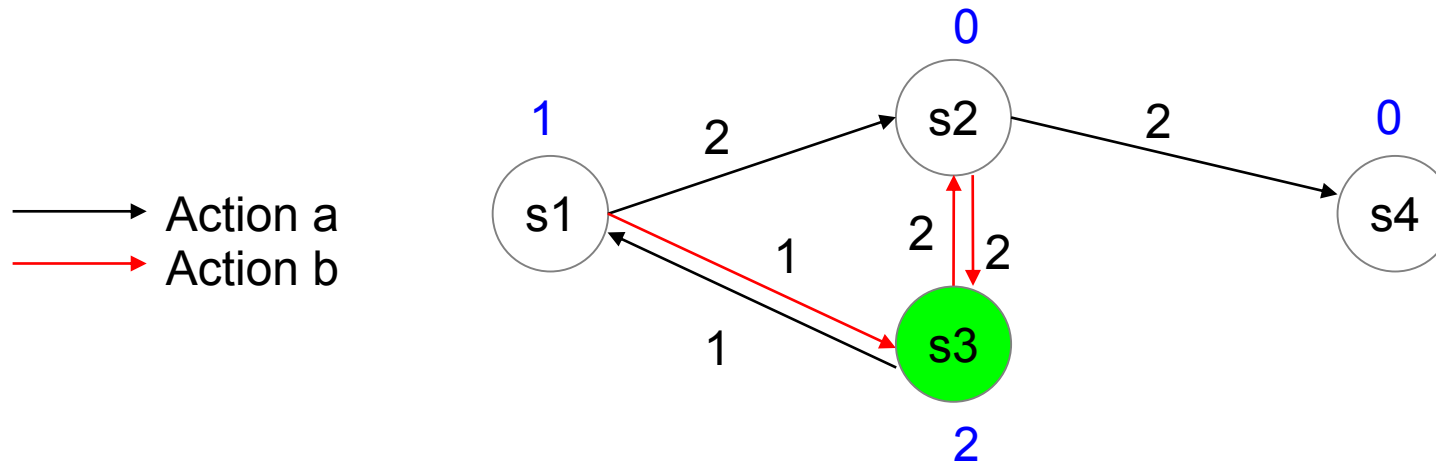
$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly
3. Update $V(s)$ to $Q(a, s)$ **LEARNING**
4. Observe resulting state s'
5. Exit: if s' is a goal, else set s to s' and go to 1

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

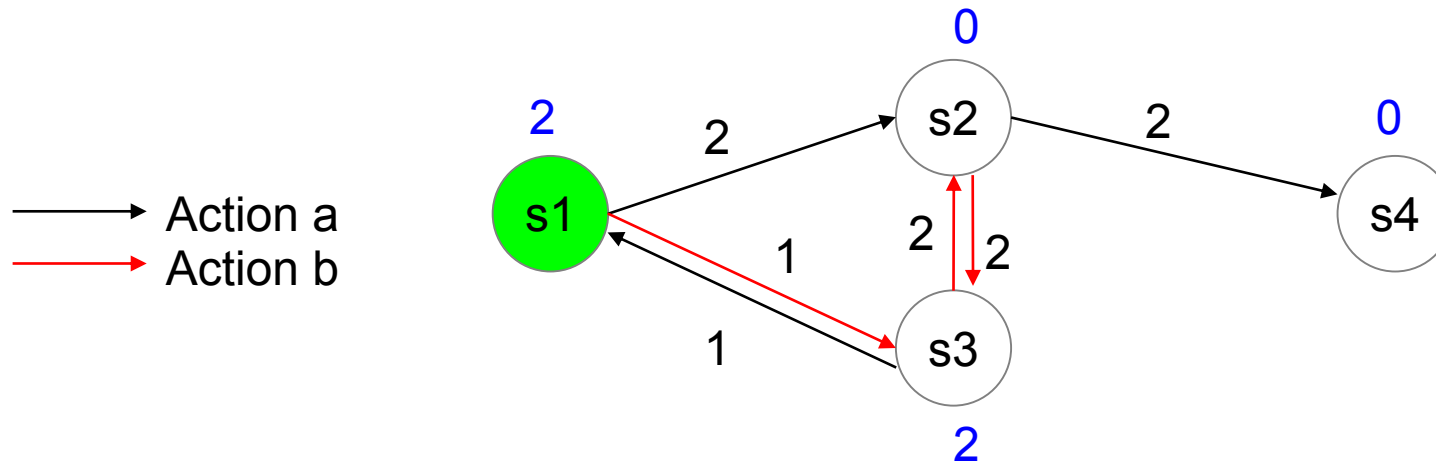
$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly
3. Update $V(s)$ to $Q(a, s)$ **LEARNING**
4. Observe resulting state s'
5. Exit: if s' is a goal, else set s to s' and go to 1

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly

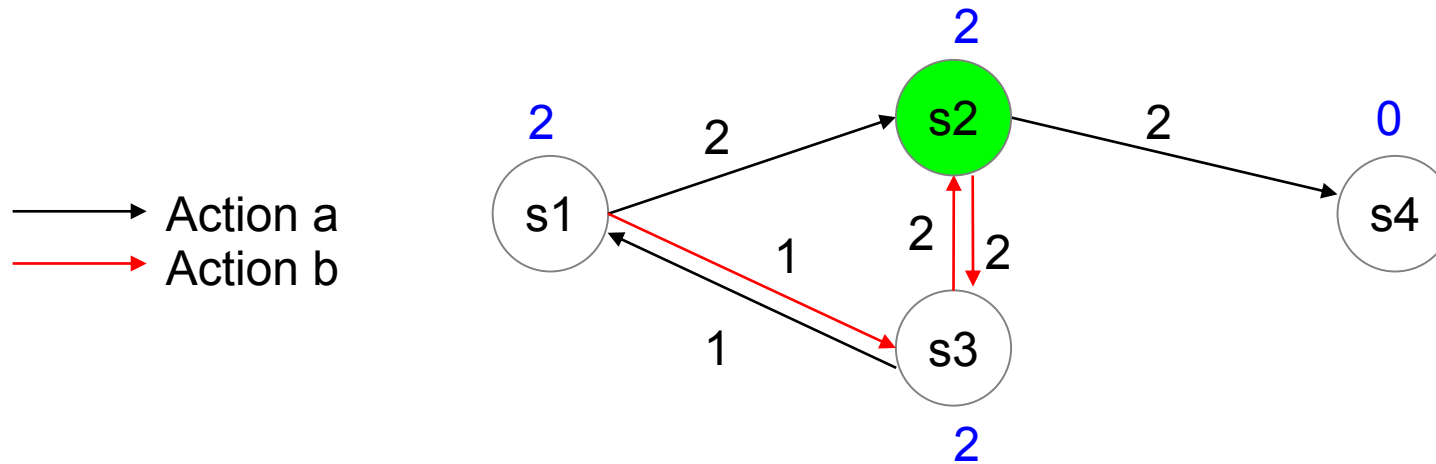
3. Update $V(s)$ to $Q(a, s)$ **LEARNING**

4. Observe resulting state s'

5. Exit: if s' is a goal, else set s to s' and go to 1

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly

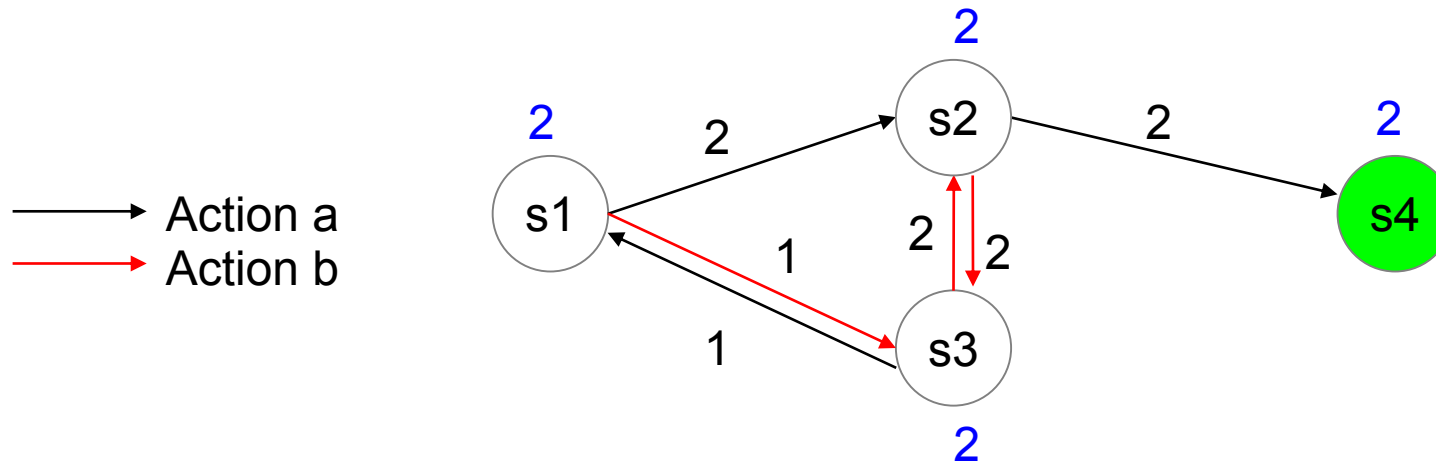
3. Update $V(s)$ to $Q(a, s)$ **LEARNING**

4. Observe resulting state s'

5. Exit: if s' is a goal, else set s to s' and go to 1

Learning Real Time Search

- Update value function for *visited* state



1. Evaluate each action a applicable in s as

$$Q(a, s) = c(a, s) + V(s_a)$$

initializing $V(s_a)$ to $h(s_a)$ when s_a is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly

3. Update $V(s)$ to $Q(a, s)$ **LEARNING**

4. Observe resulting state s'

5. Exit: if s' is a goal, else set s to s' and go to 1

Convergence of LRTA*

- Infinite cycle never exists
 - Loop is broken until value for a state reaches infinite
- If s_i chooses s_j and $v(j) = d(j)$, $v(i) = d(i)$
$$v(i) = c(i, j) + d(j) = c(i, j) + h(j) \leq c(i, i') + h(i') \leq c(i, i') + d(i')$$
 - d : distance to goal
 - i' : neighbor of i
- There is a state that $v(s) = d(s)$ in a path
 - At least $h(goal) = v(goal) = d(goal) = 0$
- **In infinite trials, every state converges to true distance**

Outline

- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- **MDP**
 - **RTDP**
- POMDP
 - RTDP-BEL
- Experimental Results
- Discussions

Markov Decision Process (MDP)

- A Markov Decision Process (MDP) model contains:
 - Set of possible world states S
 - Set of possible actions A
 - Real valued cost function $c(s, a)$
 - A description $T : S \times A \rightarrow \text{Prob}(S)$ of each action's effects in each state.
- Markov Property: the effects of an action taken in a state depend only on that state and not on the prior history.

Value Iteration for MDP

- Value function of policy π

$$V_{\pi}(s) = E \left[\sum_{\tau=t}^{\infty} \gamma^{\tau} c(s_{\tau}) \mid s_t = s, a_i = \pi(s_i) \right]$$

- Bellman equation for optimal value function

$$V_{t+1}^*(s) = \min_{a \in A(s)} \left[c(a, s) + \gamma \sum_{s' \in S} p_a(s' | s) V_t^*(s') \right]$$

- Value iteration: recursively estimating value function

$$V(s) = \min_{a \in A(s)} \left[c(a, s) + \gamma \sum_{s' \in S} p_a(s' | s) V(s') \right]$$

- Greedy policy:

$$\pi(s) = \operatorname{argmin}_{a \in A(s)} \left[c(a, s) + \gamma \sum_{s' \in S} p_a(s' | s) V(s') \right]$$

Real Time Dynamic Programming

- Incorporate discount factors in MDP

1. Evaluate each action a applicable in s as:

$$Q(a, s) = c(a, s) + \gamma \sum_{s' \in S} p_a(s'|s)V(s')$$

initializing $V(s')$ to $h(s')$ when s' is not in the table

2. Apply action a with minimum $Q(a, s)$ value, breaking ties randomly
3. Update $V(s)$ to $Q(a, s)$
4. Observe resulting state s'
5. Exit if s' is a goal, else set s to s' and go to 1

- After repeated trials, greedy policy eventually becomes optimal if $V(s)$ initialized to admissible $h(s)$

Outline

- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- MDP
 - RTDP
- POMDP
 - RTDP-BEL
- Experimental Results
- Discussions

Partially Observable MDP (POMDP)

- Since state is not observable, the agent has to make its decisions based on belief state which is a posterior probability distribution over states.
- Each value in a POMDP is a function of an entire probability distribution which are continuous.
- For finite worlds with finite state, action, and measurement spaces and finite horizons, value functions can be represented by piecewise linear functions.

Value Iteration in POMDPs

- Value function of policy π

$$V_{\pi}(b) = E \left[\sum_{\tau=t}^{\infty} \gamma^{\tau} c(a, b_{\tau}) \mid b_t = b, a_i = \pi(b_i) \right]$$

- Bellman equation for optimal value function

$$V^*(b) = \min_{a \in A(b)} \left[c(a, b) + \gamma \sum_{b' \in B} p(b' \mid b, a) V^*(b') \right]$$

- Value iteration: recursively estimating value function

$$V_t(b) = \min_{a \in A(b)} \left[c(a, b) + \gamma \sum_{b' \in B} p_a(b' \mid b) V_{t-1}(b') \right]$$

- Greedy policy:

$$\pi(b) = \operatorname{argmin}_{a \in A(b)} \left[c(a, b) + \gamma \sum_{b' \in B} p_a(b' \mid b) V(b') \right]$$

RTDP-BEL

- An action $a \in A(b)$ maps b into b_a

$$b_a(s) = \sum_{s' \in S} p_a(s' | s) b(s')$$

- The probability of observing o then is

$$b_a(o) = \sum_{s \in S} p_a(o | s) b_a(s)$$

- Thus, the new belief is

$$b_a^o(s) = \frac{p_a(o | s) b_a(s)}{b_a(o)}, \text{ when } b_a(o) \neq 0$$

- Bellman equation for belief MDPs is

$$V^*(b) = \min_{a \in A(b)} \left[c(a, b) + \gamma \sum_{o \in O} b_a(o) V^*(b_a^o) \right]$$

where,

$$c(a, b) = \sum_{s \in S} c(a, s) b(s)$$

RTDP-BEL

- RTDP in belief space B

1. Evaluate each action a applicable in b as

$$Q(a, b) = c(a, b) + \sum_{o \in O} b_a(o) V(b_a^o)$$

initializing $V(b_a^o)$ to $h(b_a^o)$ when b_a^o not in table

2. Apply action a that minimizes $Q(a, b)$ breaking ties randomly

3. Update $V(b)$ to $Q(a, b)$

4. Observe o

5. Compute b_a^o

6. Exit if b_a^o is a goal (belief) state, else set b to b_a^o and go to 1

- Goal belief states are $b_G(s) = 0 \quad \forall s \notin G$

Implementation

- Belief discretization
 - Transforming the infinite state space to finite
 - Resolution parameter r
 - Makes hash table smaller and learning faster
- Max step for each trial
- Sparse presentation to represent belief states to decrease complexity to $|S|^2$
- Complexity of RTDP-BEL is $|S|^2 \times |O|$
 - But convergence depends on quality of h_{MDP}

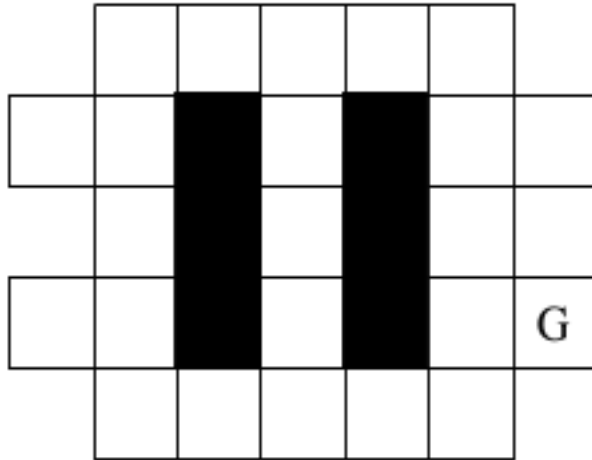
Summary

- Real Time Search
 - Heuristic function invariant
- Learning Real Time Search
 - Replace heuristic function with value function
 - Update value function for visited states
- Real Time Dynamic Programming (RTDP)
 - Incorporate discount factor in MDP
- RTDP-BEL
 - Incorporate observation in POMDP

Outline

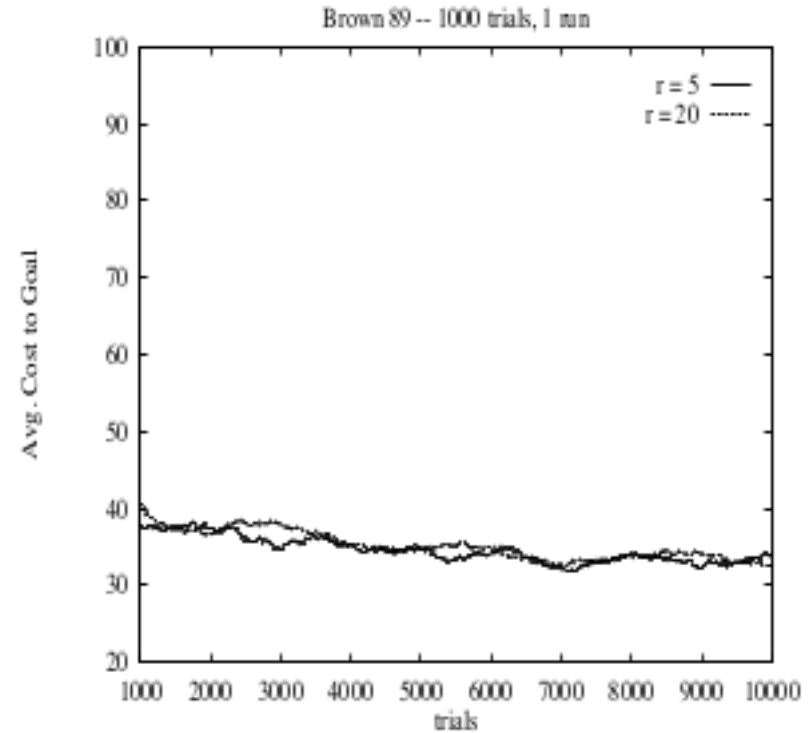
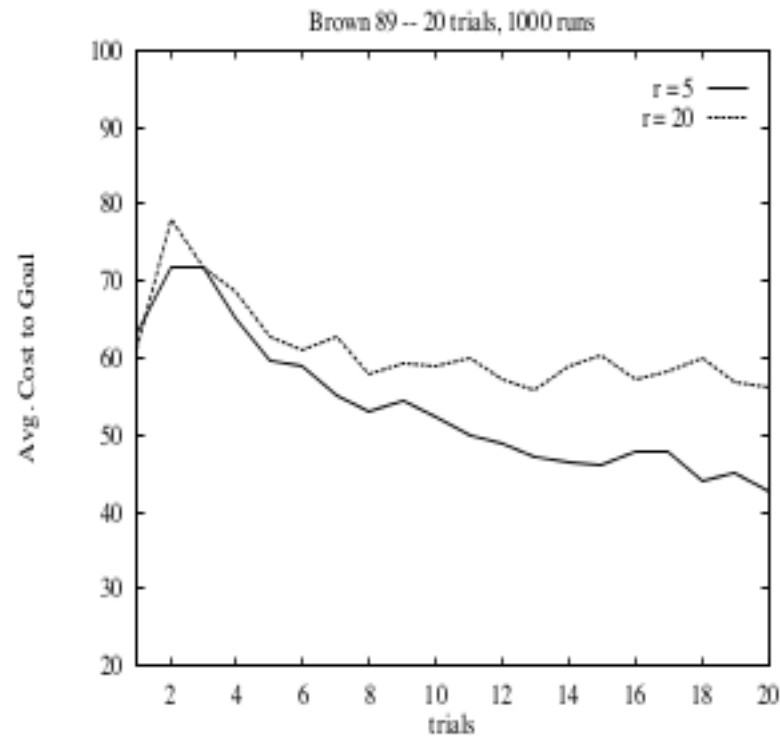
- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- MDP
 - RTDP
- POMDP
 - RTDP-BEL
- **Experimental Results**
- Discussions

Medium Problems



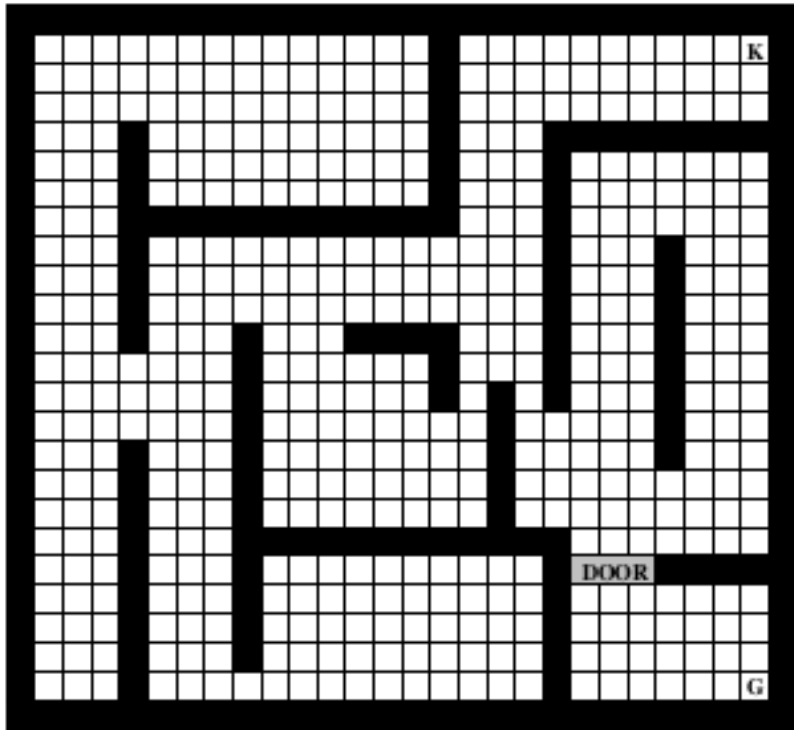
- Start : random point
- Goal : G
- 89 states (4 orientations in 22 rooms + goal)
- 17 observations (placement of walls in 4 orientations)
- 5 actions (4 orientations + stay)
- Observations and actions with accuracy of 70%

Medium Problems



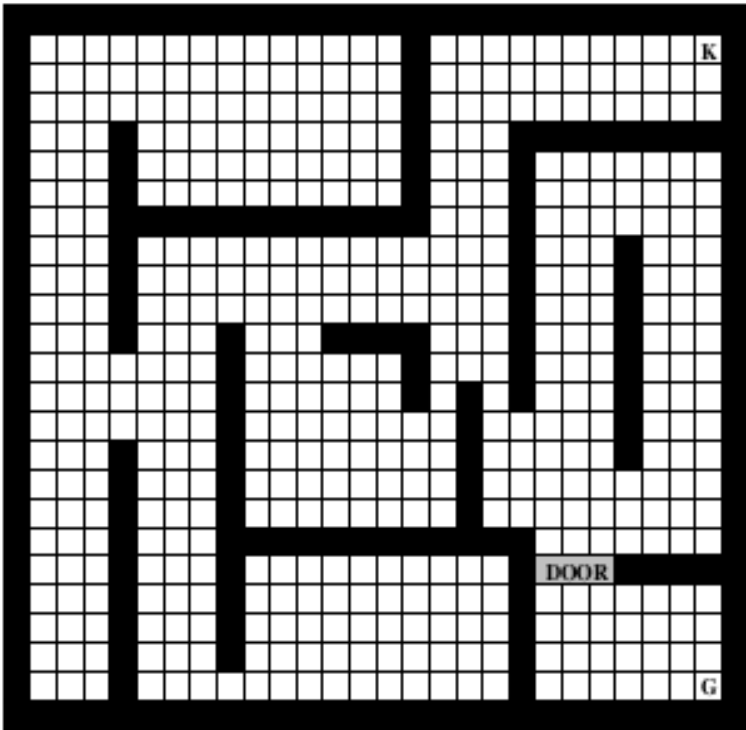
- r : Discretization number for belief state.
Example: $r=5$, probability is rounded to 0, 0.2, 0.4, 0.6, 0.8, 1
- Average cost for human expert is 29

Large problems



- Start : random point
- K : key
- G : goal
- 989 states
- No exact information about position & possession of the key

Large problems



- Parameter
 - belief state resolution
 $r = 20$
 - discount factor = 0.95
- Results
 - Cost to the goal value 68.44 for first trial (average over 1000 runs)
 - Cost to the goal value 67 after 10000 trials (average over last 2000 trials)

Outline

- Two paradigms- offline vs. online
- Classical scenario
 - RTS
 - LRTA*
- MDP
 - RTDP
- POMDP
 - RTDP-BEL
- Experimental Results
- **Discussions**

Discussions

- Restricting updates to the states generated by the stochastic simulation of the greedy policy
- Competitive with point based algorithms
- RTDP-BEL is not optimal
 - No theoretical guarantee of convergence
 - Limited resolution introduces errors
- Fails for large spaces, where search for information precedes search for goal
- Not applicable to discounted POMDPs

Discussions

- Restricting updates to the states generated by the stochastic simulation of the greedy policy
- Competitive with point based algorithms
- RTDP-BEL is not optimal
 - No theoretical guarantee of convergence
 - Limited resolution introduces errors
- Fails for large spaces, where search for information precedes search for goal
- Not applicable to discounted POMDPs

Questions?

Thank you!!