Planning and Optimization

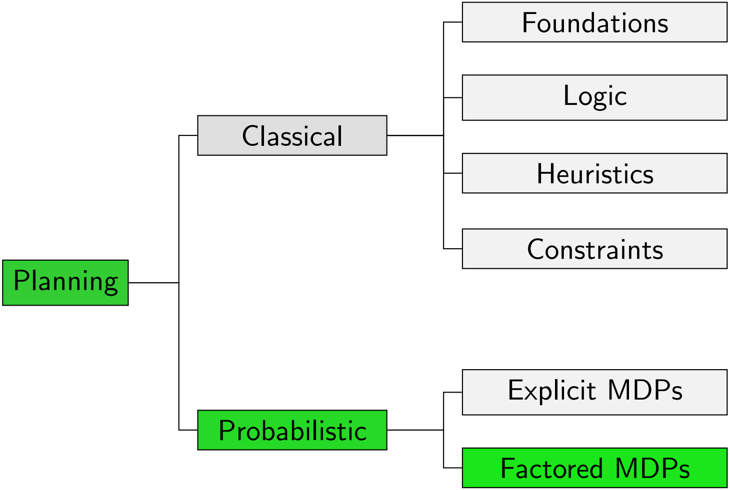
G2. AO* & LAO*

Malte Helmert and Thomas Keller

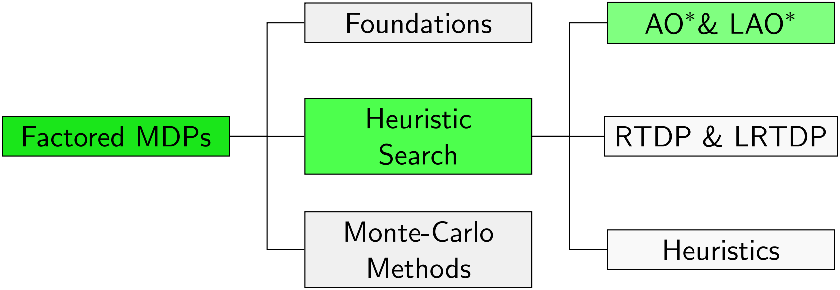
Universität Basel

December 4, 2019

Content of this Course



Content of this Course: Factored MDPs



Heuristic Search

Reminder: Heuristic Search

Heuristic Search Algorithms

Heuristic search algorithms use heuristic functions
to (partially or fully) determine the order of node expansion.

(From Lecture 15 of the AI course last semester)

Reminder: Best-first Search

Best-first Search

A best-first search is a heuristic search algorithm
that evaluates search nodes with an evaluation function f
and always expands a node n with minimal $f(n)$ value.

(From Lecture 15 of the AI course last semester)

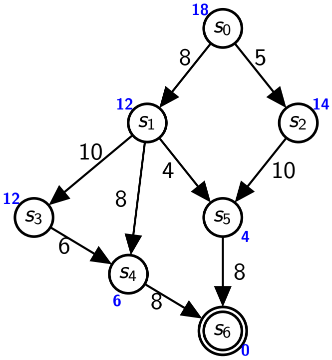
Reminder: A*Search

A*Search

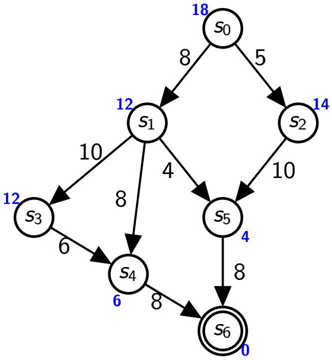
A* is the best-first search algorithm with evaluation function $f(n) = g(n) + h(n.\text{state})$.

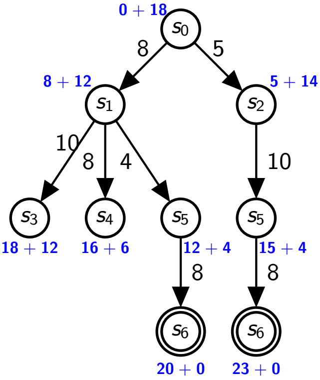
(From Lecture 16 of the AI course last semester)

A* Search (With Reopening): Example



A^* Search (With Reopening): Example





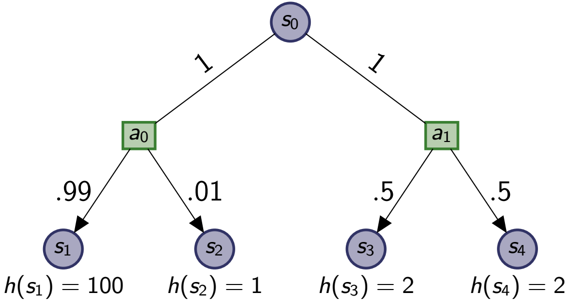
Motivation

From A*to AO*

- equivalent of A* for (acyclic) SSPs is AO*
- the generalization is not straightforward:
  - A* always expands most promising state
  - it uses $g(n)$ as cost from root n_0 to n
  - Can we replace this in SSPs with expected cost from n_0 to n?

Expected Cost to Reach State

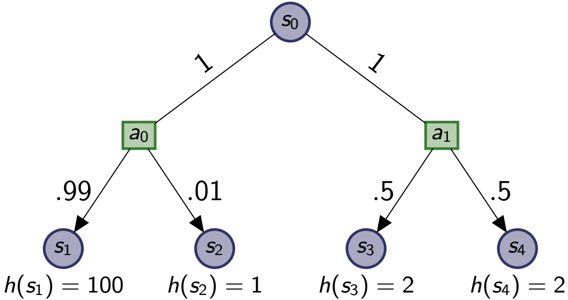
Consider the following expansion of state s_0:



What is the expected cost to reach each leaf?

Expected Cost to Reach State

Consider the following expansion of state s_0:



What is the expected cost to reach each leaf?

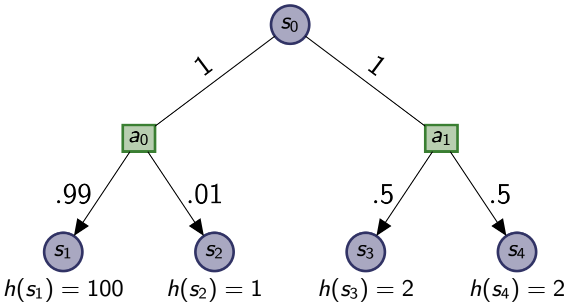
Answer: undefined, as neither of them is reached with probability 1

From A*to AO*

- equivalent of A* for (acyclic) SSPs is AO*
- the generalization is not straightforward:
  - A* always expands most promising state
  - it uses $g(n)$ as cost from root n_0 to n
  - Can we replace this in AO* with expected cost from n_0 to n?
  - Is expected cost from n_0 to n given n is reached an alternative?

Expected Cost to Reach State Given It Is Reached

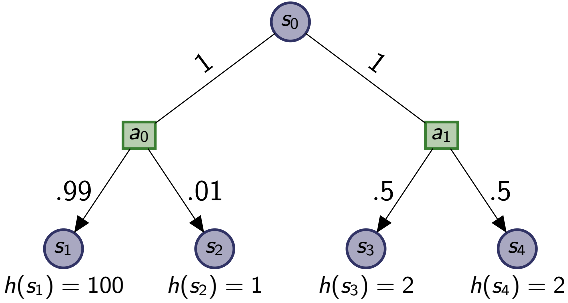
Consider the following expansion of state s_0:



What is the expected cost to reach each leaf given it's reached?

Expected Cost to Reach State Given It Is Reached

Consider the following expansion of state s_0:

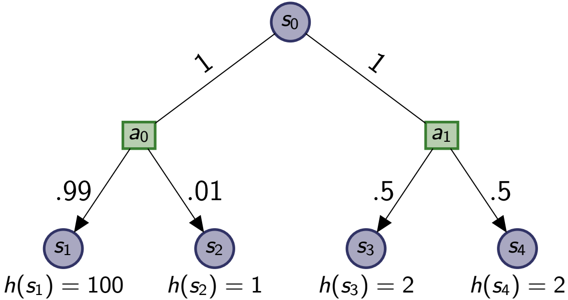


What is the expected cost to reach each leaf given it's reached?
Answer: 1 for all, so s_2 is expanded due to minimal f value

Is expanding a successor of a_0 a "most promising" choice?

Expected Cost to Reach State Given It Is Reached

Consider the following expansion of state s_0:



What is the expected cost to reach each leaf given it's reached?
Answer: 1 for all, so s_2 is expanded due to minimal f value

Is expanding a successor of a_0 a "most promising" choice?
Answer: No, because it's likely that s_1 is reached if a_0 is applied.

Expansion in Best Solution Graph

Instead of expanding the state with minimal f-value, AO* exploits a different idea:

- AO* keeps track of best solution graph
- AO* expands a state that can be reached from s_0 by only applying greedy actions
- $\Rightarrow$ no g-value equivalent required

Outlook

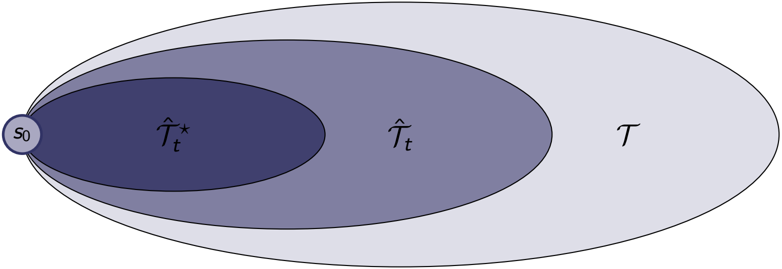
- Equivalent version of A^* built on this idea can be derived
  $\Rightarrow$ A^* with backward induction
- Since change is non-trivial, we focus on A^* variant now
- and generalize later to acyclic SSPs (AO^*)
- and SSPs with cycles (LAO^*)

A* with Backward Induction

Transition Systems

A* with backward induction distinguishes three transition systems:

- The transition system $\mathcal{T} = \langle S, L, c, T, s_0, S^\star \rangle$
  $\Rightarrow$ given implicitly
- The explicated graph $\hat{\mathcal{T}}_t = \langle \hat{S}_t, L, c, \hat{T}_t, s_0, S^\star \rangle$
  $\Rightarrow$ the part of $\mathcal{T}$ explicitly considered during search
- The partial solution graph $\hat{\mathcal{T}}_t^\star = \langle \hat{S}_t^\star, L, c, \hat{T}_t^\star, s_0, S^\star \rangle$
  $\Rightarrow$ The part of $\hat{\mathcal{T}}_t$ that contains best solution



Explicated Graph

- Expanding a state s at time step t explicates all successors $s' \in \mathrm{succ}(s)$ by adding them to explicated graph:

$$\hat{\mathcal{T}}_t = \langle \hat{S}_{t-1} \cup \mathrm{succ}(s), L, c, \hat{T}_{t-1} \cup \{\langle s, \ell, s' \rangle \in T\}, s_0, S^\star \}$$

- Each explicated state is annotated with state-value estimate $\hat{V}_t(s)$ that describes estimated cost to a goal at time step t
- When state s' is explicated and $s' \notin \hat{S}_{t-1}$, its state-value estimate is initialized to $\hat{V}_t(s') := h(s')$
- We call leaf states of $\hat{\mathcal{T}}_t$ fringe states

Partial Solution Graph

- The partial solution graph $\hat{\mathcal{T}}_t^\star$ is the subgraph of $\hat{\mathcal{T}}_t$ that is spanned by the smallest set of states $\hat{S}_t^\star$ that satisfies:
  - $s_0 \in \hat{S}_t^\star$
  - if $s \in \hat{S}_t^\star$, $s' \in \hat{S}_t$ and $\langle s, a_{\hat{V}_t(s)}(s), s' \rangle \in \hat{T}_t$, then s' in $\hat{S}_t^\star$
- The partial solution graph forms a sequence of states $\langle s_0, \ldots, s_n \rangle$, starting with the initial state s_0 and ending in the greedy fringe state s_n

Backward Induction

- A* with backward induction does not maintain static open list
- State-value estimates determine partial solution graph
- Partial solution graph determines which state is expanded
- (Some) state-value estimates are updated in time step t by backward induction:

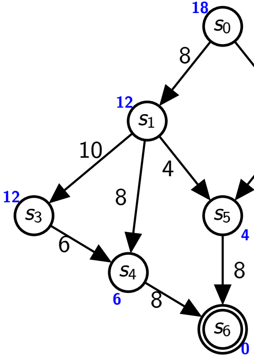
$$\hat{V}_t(s) = \min_{\langle s,\ell,s'\rangle \in \hat{T}_t(s)} \left(c(\ell) + \hat{V}_t(s') \right)$$

A* with backward induction

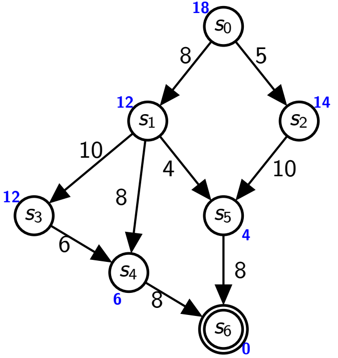
A* with backward induction for classical planning task $\mathcal{T}$

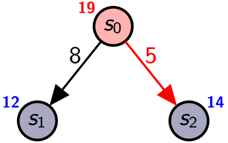
explicate s_0
while greedy fringe state $s \notin S_\star$:
  expand s
  perform backward induction of states in $\hat{\mathcal{T}}^\star_{t-1}$ in reverse order
return $\hat{\mathcal{T}}^\star_t$

A* with backward induction

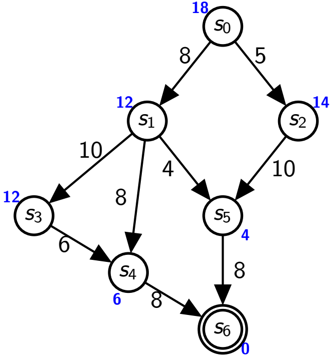


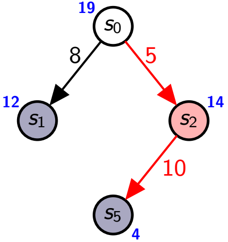
A* with backward induction



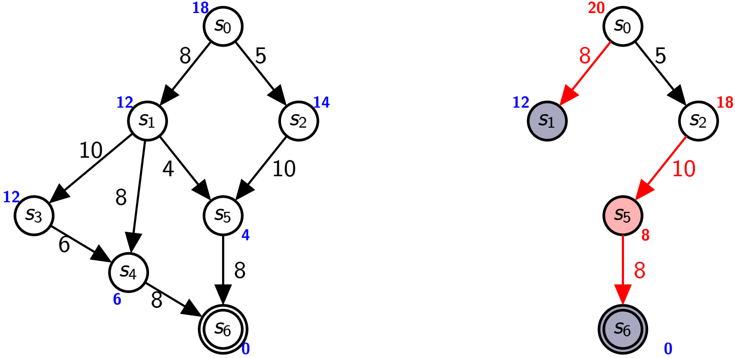


A* with backward induction

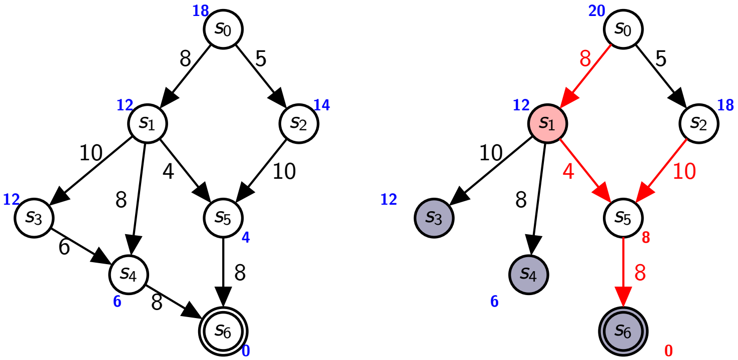




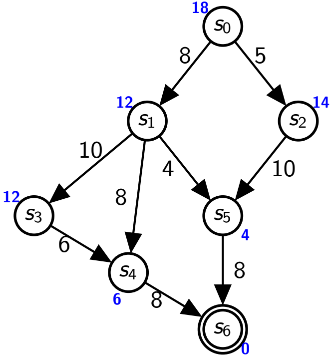
A* with backward induction

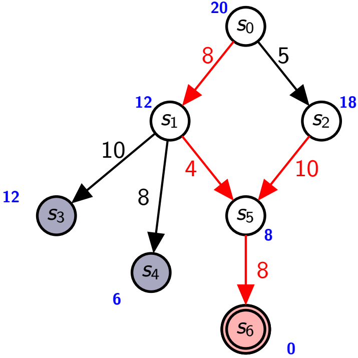


A* with backward induction



A* with backward induction





Equivalence of A^* and A^* with Backward Induction

Theorem

A^* and A^* with Backward Induction expand the same set of states if run with identical admissible heuristic h and identical tie-breaking criterion.

Proof Sketch.

The proof shows that

- the fringe states of the explicated graph A^* with backward induction correspond to the states in the open list of A^*
- the f-value of the greedy fringe state of A^* with backward induction is minimal among all fringe states

AO*

From A* with Backward Induction to AO*

A* with backward induction already very similar to AO*, only support for uncertain outcomes missing. Need to adapt:

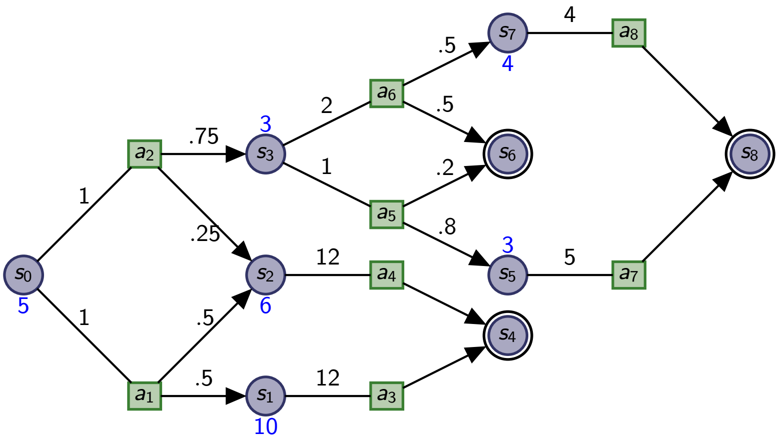
- Which states are explicated upon expansion?
  ⇒ all outcomes
- Which form does the partial solution graph have?
  ⇒ a partial acyclic policy
- Which state is selected for expansion?
  ⇒ any greedy fringe state
  (e.g., the state that is most likely reached)
- How are states updated?
  ⇒ by applying Bellman equation as update rule
- When does the algorithm terminate?
  ⇒ when all states in the greedy fringe are goal states

AO*

AO* for acyclic SSP $\mathcal{T}$

explicate s_0
while there is a greedy fringe state not in $S_\star$:
    select a greedy fringe state $s \notin S_\star$
    expand s
    perform Bellman backups of states in $\hat{\mathcal{T}}^\star_{t-1}$ in reverse order
return $\hat{\mathcal{T}}^\star_t$

AO*: Example (Blackboard)



$h(s) = 0$ for goal states, otherwise in blue above or below s

Theoretical properties

Theorem

Using an admissible heuristic, AO converges to an optimal solution without (necessarily) explicating all states.*

Proof omitted.

LAO*

LAO*

- A* with backward induction finds sequential solutions (a plan) in classical planning tasks
- AO* finds acyclic solutions with branches (an acyclic policy) in acyclic SSPs
- LAO* is the generalization of AO* to cyclic solutions in cyclic SSPs

From AO* to LAO*

- From plans to acyclic policies, we only changed backup procedure to consider transition probabilities
- When solutions may be cyclic, we cannot order states in a way that guarantees that all successors have been updated before
- We need an iterative process to perform backups
- the original algorithm of Hansen & Zilberstein (1998) uses Policy Iteration

LAO^*

LAO^* for SSP $\mathcal{T}$

explicate s_0
while there is a greedy fringe state not in $S_\star$:
    select a greedy fringe state $s \notin S_\star$
    expand s
    perform policy iteration in $\hat{\mathcal{T}}_t$
return $\hat{\mathcal{T}}_t^\star$

LAO*: Optimizations

Several optimizations for LAO* have been proposed:

- Use Value Iteration instead of PI
- Terminate VI when the partial solution graph changes
- Expand all states in greedy fringe before backup
- Order states (arbitrarily within cycles) and use backward induction for updates

⇒ last two combine to famous variant iLAO*

Theoretical properties

Theorem

Using an admissible heuristic, LAO* *converges to an optimal solution without (necessarily) explicating all states.*

Proof omitted.

Summary

Summary

- Non-trivial to generalize A^* to probabilistic planning
- For better understanding of AO^*, we change A^* towards AO^*
- Derived A^* with backward induction, which is similar to AO^*
- and expands identical states as A^*
- AO^* finds optimal solutions for acyclic SSPs
- LAO^* finds optimal solutions for SSPs