

Planning and Optimization

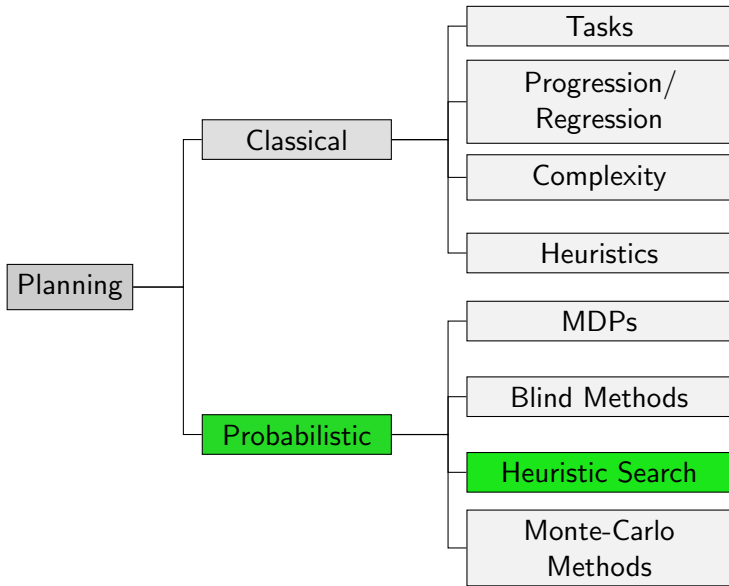
G2. Heuristic Search: AO* & LAO* Part II

Gabriele Röger and Thomas Keller

Universität Basel

December 3, 2018

Content of this Course



AO*

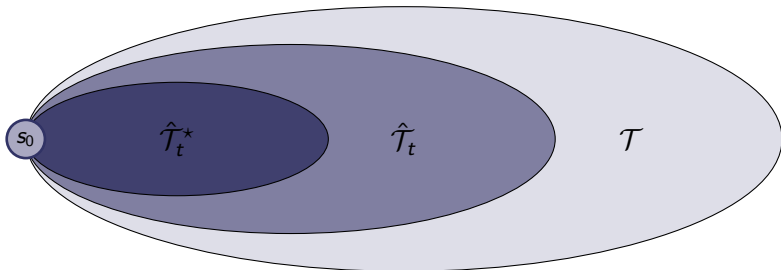
From A* with Backward Induction to AO*

- A* with backward induction already very similar to AO*
- Support for **uncertain outcomes** missing
- We focus on SSPs in these slides
- Adaption to FH-MDPs simple
 - **Careful**: admissible heuristic in reward setting **must not underestimate** true reward
- Still two steps ahead:
 - restrict to **acyclic probabilistic tasks** \Rightarrow AO*
 - allow **general probabilistic tasks** \Rightarrow LAO*

Transition Systems

AO* distinguishes three transition systems:

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



Explicated Graph

- Expanding a state s at time step t explicates all **outcomes** $s' \in \text{succ}(s, \ell)$ for all $\ell \in L(s)$ by adding them to explicated graph:

$$\hat{\mathcal{T}}_t = \langle \hat{\mathcal{S}}_{t-1} \cup \text{succ}(s), L, c, \hat{\mathcal{T}}_t, s_0, S^* \rangle,$$

where $\hat{\mathcal{T}}_t = \hat{\mathcal{T}}_{t-1}$ except that $\hat{\mathcal{T}}_t(s, \ell, s') = T(s, \ell, s')$ for all $\ell \in L(s)$ and $s' \in \text{succ}(s, \ell)$

- Explicated states are annotated with **state-value estimate** $\hat{V}_t(s)$ that describes **estimated expected cost to goal** at step t
- When state s' is explicated and $s' \notin \hat{\mathcal{S}}_{t-1}$, 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** $\hat{\mathcal{T}}_t^*$ is the subgraph of $\hat{\mathcal{T}}_t$ that is spanned by the **smallest set** of states \hat{S}_t^* that satisfies:
 - $s_0 \in \hat{S}_t^*$
 - if $s \in \hat{S}_t^*$, $s' \in \hat{S}_t$ and $\hat{T}_t(s, a_{\hat{V}_t}(s), s') > 0$, then $s' \in \hat{S}_t^*$
- The partial solution graph forms a **partial acyclic policy** defined in the initial state s_0 and all **non-leaf states** that can be reached by its execution
- Leaf states that can be reached by the policy described by the partial solution graph are the **states in the greedy fringe**

Bellman backups

- AO* does not maintain **static open list**
- **State-value estimates** determine **partial solution graph**
- **Partial solution graph** determines which state is a **candidate for expansion**
Different strategies to **select among candidates** exist
- (Some) state-value estimates are **updated** in time step t by **Bellman backups**:

$$\hat{V}_t(s) = \min_{l \in L} c(l) + \sum_{s' \in \hat{S}_t} \hat{T}_t(s, l, s') \cdot \hat{V}_t(s')$$

AO*

AO* for acyclic SSP \mathcal{T}

explicate s_0

while there is a greedy fringe state not in S_* :

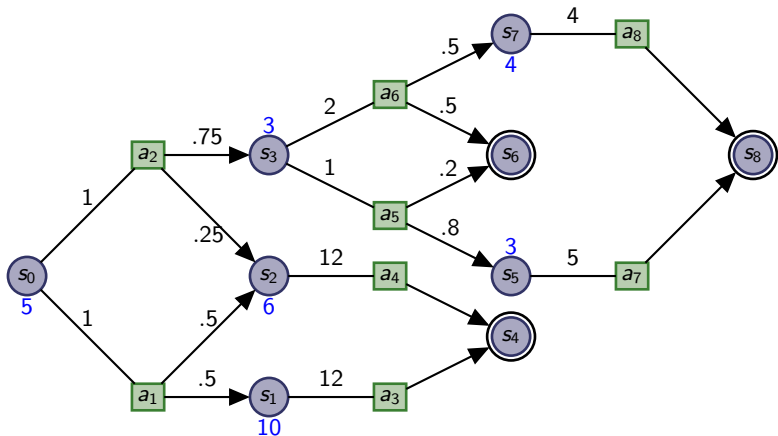
 select a greedy fringe state $s \notin S_*$

 expand s

 perform Bellman backups of states in $\hat{\mathcal{T}}_{t-1}^*$ in reverse order

return $\hat{\mathcal{T}}_t^*$

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

LAO*

- From plans to acyclic policies, we only changed backup procedure from **backward induction** to **Bellman backups**
- When solutions may be cyclic, we cannot perform **updates in reverse order**

LAO*

- From plans to acyclic policies, we only changed backup procedure from **backward induction** to **Bellman backups**
- When solutions may be cyclic, we cannot perform **updates in reverse order**
- Bellman backups are essentially **acyclic version of value iteration**

LAO*

- From plans to acyclic policies, we only changed backup procedure from **backward induction** to **Bellman backups**
- When solutions may be cyclic, we cannot perform **updates in reverse order**
- Bellman backups are essentially **acyclic version of value iteration**
- replacing Bellman backups with **value iteration** is LAO* variant
- the original algorithm of Hansen & Zilberstein (1998) uses **policy iteration** instead

LAO*

LAO* for SSP \mathcal{T}

explicate s_0

while there is a greedy fringe state not in S_* :

 select a greedy fringe state $s \notin S_*$

 expand s

 perform policy iteration in $\hat{\mathcal{T}}_t$

return $\hat{\mathcal{T}}_t^*$

LAO*: Optimizations

Several optimizations for LAO* have been proposed:

- Use **value iteration** instead of policy iteration
- **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

- AO* finds **optimal solutions** for acyclic SSPs
- LAO* finds **optimal solutions** for SSPs
- Both algorithms differ from A* with backward induction in way **backups** are performed
- Unlike previous optimal algorithms, both are able to find optimal solution **without explicating all states**