Module 9
LAO*

CS 886 Sequential Decision Making and Reinforcement Learning

University of Waterloo

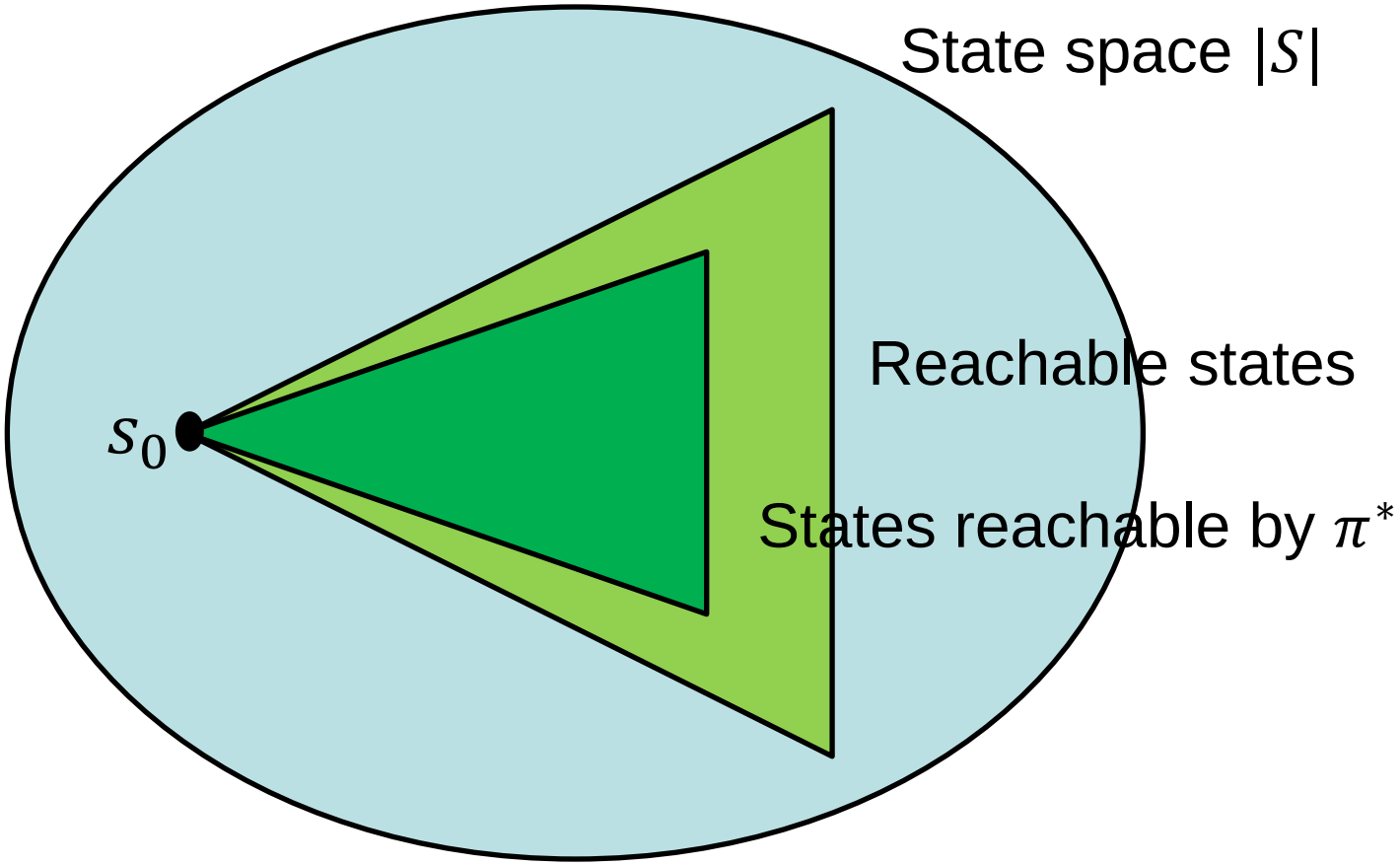
Large State Space

- Value Iteration, Policy Iteration and Linear Programming
  - Complexity at least quadratic in $|S|$
- Problem: $|S|$ may be very large
  - Queuing problems: infinite state space
  - Factored problems: exponentially many states

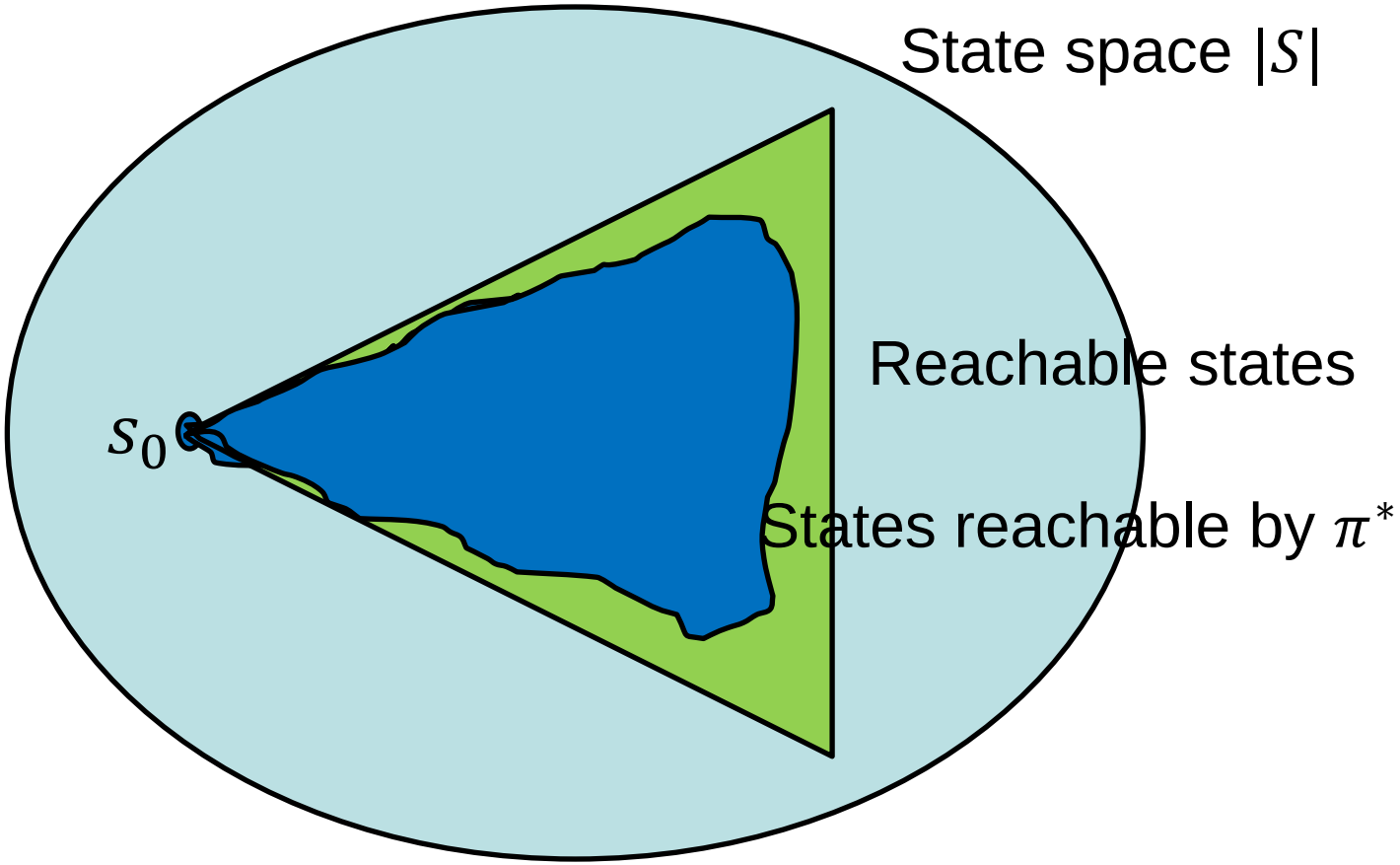
Mitigate Size of State Space

- Two ideas:
- Exploit initial state
  - Not all states are reachable
- Exploit heuristic h
  - approximation of optimal value function
  - usually an upper bound $h(s) \geq V^*(s)\ \forall s$

State Space



State Space



LAO* Algorithm

- Related to
  - A*: path heuristic search
  - AO*: tree heuristic search
  - LAO*: cyclic graph heuristic search

- LAO* alternates between
  - State space expansion
  - Policy optimization
    - value iteration, policy iteration, linear programming

Terminology

- S: state space
- $S_E \subseteq S$: envelope
  - Growing set of states
- $S_T \subseteq S_E$: terminal states
  - States whose children are not in the envelope
- $S_{s_0}^{\pi} \subseteq S_E$: states reachable from s_0 by following π
- $h(s)$: heuristic such that $h(s) \geq V^*(s)\ \forall s$
  - E.g., $h(s) = \max_{s,a} R(s,a)/(1-\gamma)$

LAO* Algorithm

LAO*(MDP, heuristic h)

$S_E \leftarrow \{s_0\}, S_T \leftarrow \{s_0\}$

Repeat

Let $R_E(s,a) = \begin{cases} h(s) & s \in S_T \\ R(s,a) & \text{otherwise} \end{cases}$

Let $T_E(s'|s,a) = \begin{cases} 0 & s \in S_T \\ \Pr(s'|s,a) & \text{otherwise} \end{cases}$

Find optimal policy π for $\langle S_E, R_E, T_E \rangle$

Find reachable states $S_{s_0}^{\pi}$

Select reachable terminal states $\{s_1, \ldots, s_k\} \subseteq S_{s_0}^{\pi} \cap S_T$

$S_T \leftarrow (S_T \setminus \{s_1, \ldots, s_k\}) \cup (children(\{s_1, \ldots, s_k\}) \setminus S_E)$

$S_E \leftarrow S_E \cup children(\{s_1, \ldots, s_k\})$

Until $S_{s_0}^{\pi} \cap S_T$ is empty

Efficiency

Efficiency influenced by

1. Choice of terminal states to add to envelope

2. Algorithm to find optimal policy
   - Can use value iteration, policy iteration, modified policy iteration, linear programming
   - Key: reuse previous computation
     - E.g., start with previous policy or value function at each iteration

Convergence

- Theorem: LAO* converges to the optimal policy
- Proof:
  - Fact: At each iteration, the value function V is an upper bound on V^* due to the heuristic function h
  - Proof by contradiction: suppose the algorithm stops, but π is not optimal.
    - Since the algorithm stopped, all states reachable by π are in $S_E \setminus S_T$
    - Hence, the value function V is the value of π and since π is suboptimal then $V < V^*$, which contradicts the fact that V is an upper bound on V^*

Summary

- LAO*
  - Extension of basic solution algorithms (value iteration, policy iteration, linear programming)
  - Exploit initial state and heuristic function
  - Gradually grow an envelope of states
  - Complexity depends on # of reachable states instead of size of state space