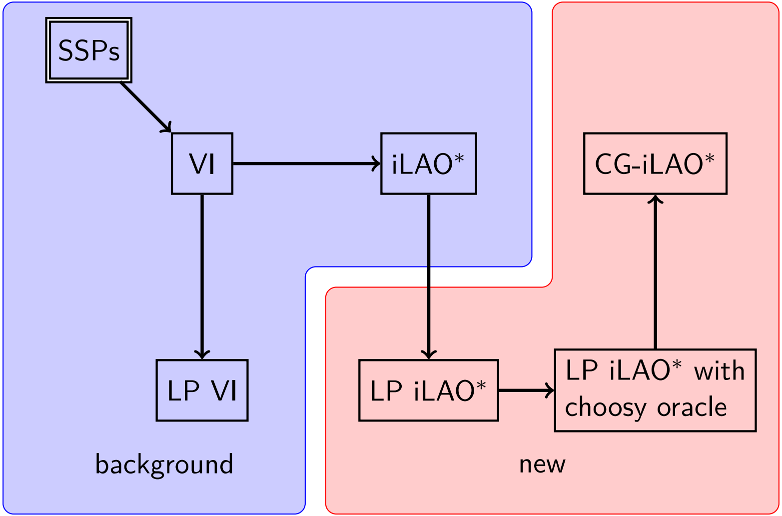
Efficient Constraint Generation for Stochastic Shortest Path Problems

Johannes Schmalz, Felipe Trevizan

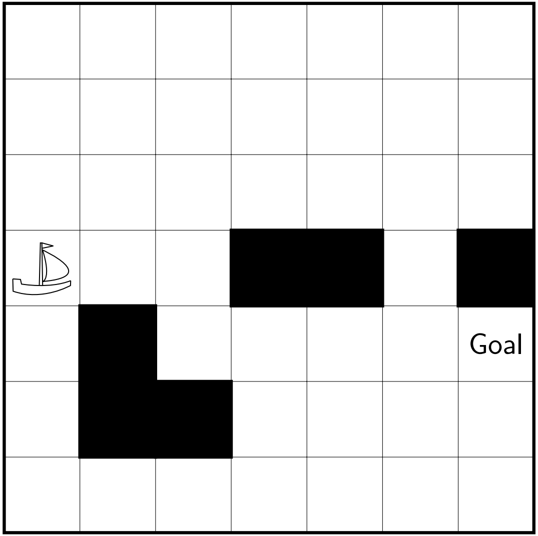
Australian National University

February 24, 2024

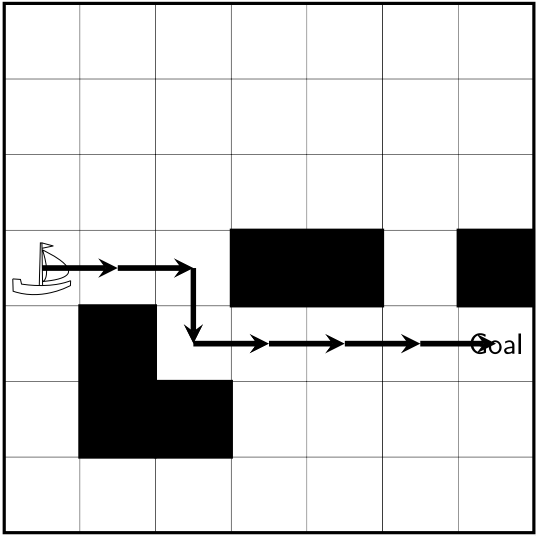
johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au


SSPs
VI
iLAO*
CG-iLAO*
LP VI
LP iLAO*
LP iLAO* with choosy oracle
background
new

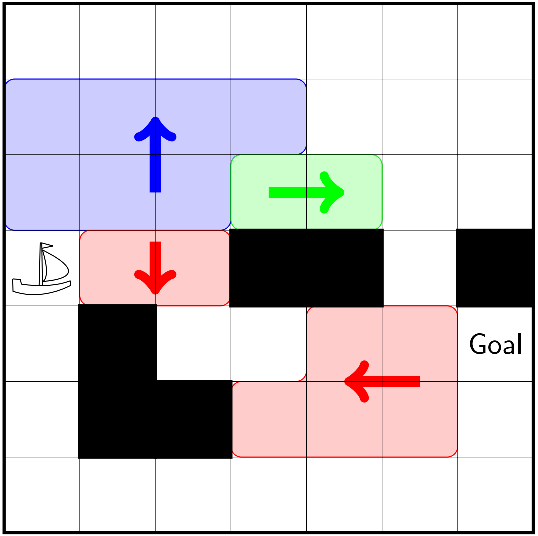
Deterministic Shortest Path Problem



Deterministic Shortest Path Problem

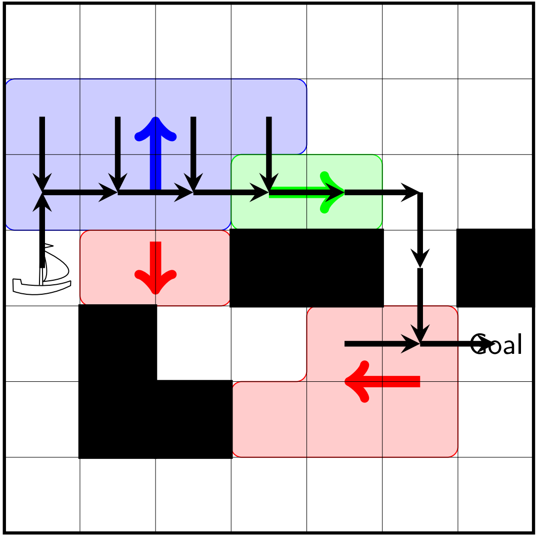


Stochastic Shortest Path Problem (SSP)



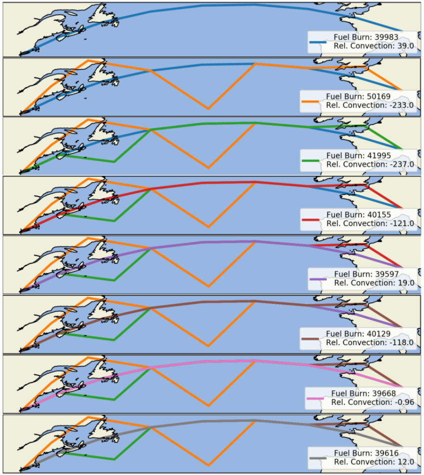
Windy zones have 50% chance to push ship in direction of arrow.

Stochastic Shortest Path Problem (SSP)



Windy zones have 50% chance to push ship in direction of arrow.

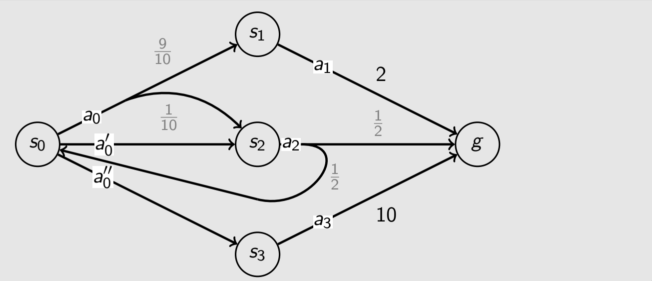
Constrained SSP (**not in this talk**, only for context!)



Geisser et al. (2020)

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Graph



Probabilistic PDDL

```
(:action move-ship
:parameters(...) :precondition(...)
:effect(probabilistic 0.9 (...)
                      0.1 (...)))
```

Tuple

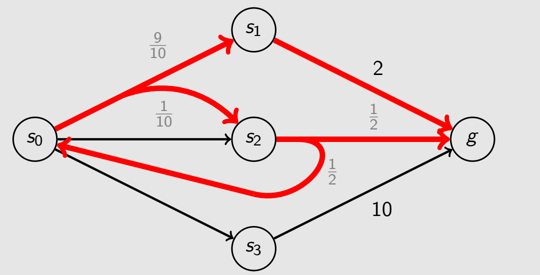
SSP $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C \rangle$

Probability Transition Matrix

$$\begin{array}{c|ccccc|}
 & s_0 & s_1 & s_2 & s_3 & g \\
s_0, a_0 & & \frac{9}{10} & \frac{1}{10} & & \\
s_0, a_0' & & & 1 & & \\
s_0, a_0'' & & & & 1 & \\
s_1, a_1 & & & & & 1 \\
s_2, a_2 & \frac{1}{2} & & & & \frac{1}{2} \\
s_3, a_3 & & & & & 1
\end{array}$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Policy



Assumptions about the SSP

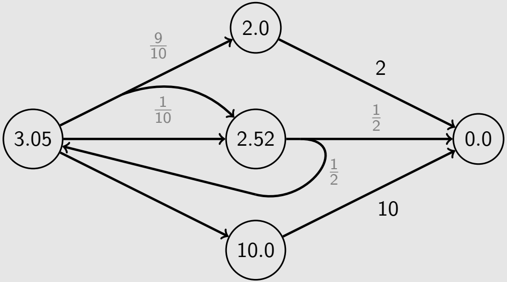
❶ can reach the goal from any state (reachability)

❷ policy that reaches goal with prob. < 1 (improper) has infinite cost

Properties of Optimal Policy

- defined for all states reachable under policy (closed)
- reaches the goal with prob. 1 (proper)
- can have cycles

Optimal Cost-to-go (V^*)



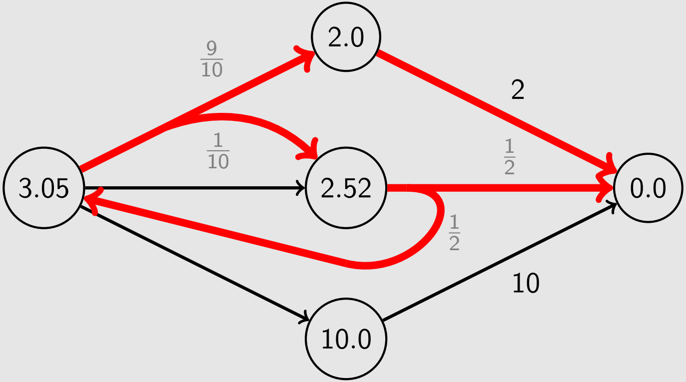
Bellman Equations

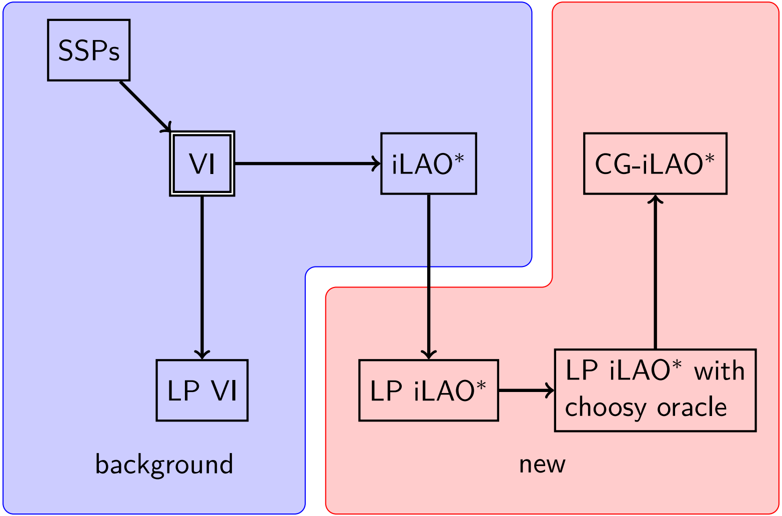
$$V(g) = 0 \qquad \forall \text{ goals } g \in \mathrm{G}$$

$$V(s) = \min_{\text{actions } a} \underbrace{C(a) + \sum_{\text{states } s'} P(s'|s,a) \cdot V(s')}_{Q(s,a)} \qquad \forall \text{ states } s \notin \mathrm{G}$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Greedy Policy is Optimal with Optimal Cost-to-go




SSPs
VI
iLAO*
CG-iLAO*
LP VI
LP iLAO*
LP iLAO* with choosy oracle
background
new

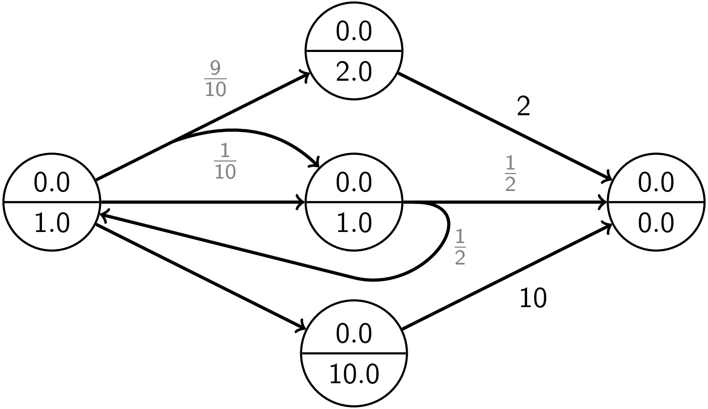
Bellman backup

$$V_{i+1}(s) \leftarrow \min_{\text{actions } a} \underbrace{C(a) + \sum_{\text{states } s'} P(s'|s,a) \cdot V_i(s')}_{Q_i(s,a)} \quad \forall \text{ states } s \notin \text{G}$$

Value Iteration (VI) Bellman (1957)

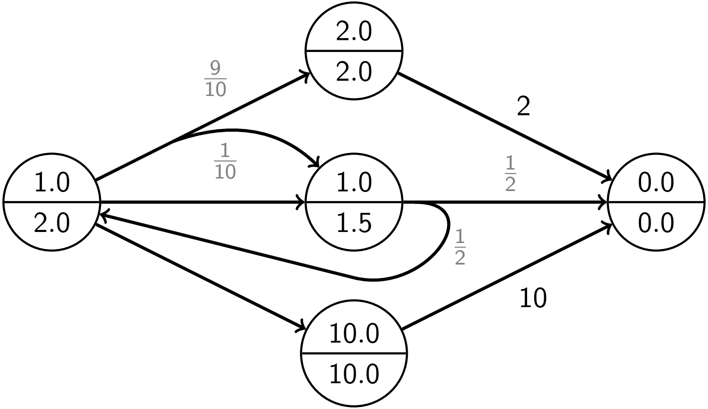
- Start with some V_0
- Loop: apply Bellman backups
- Stop when changes are small

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

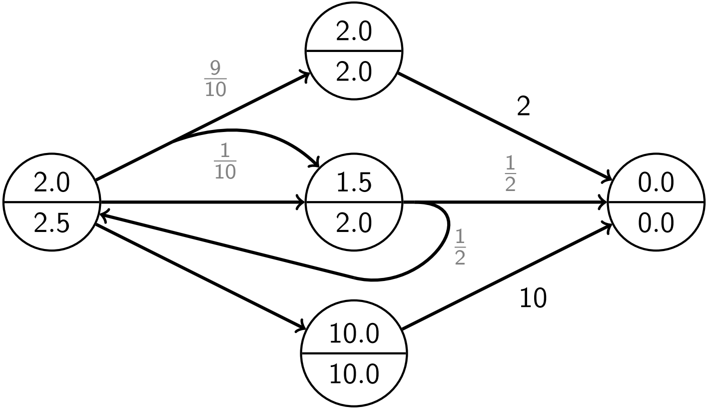


Top: V_0, Bottom: V_1

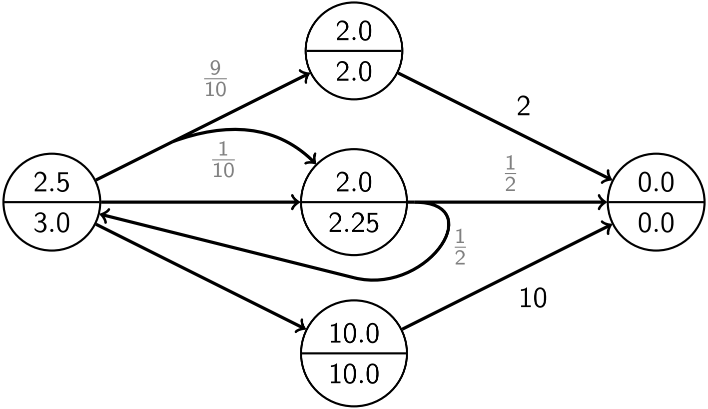
johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au



Top: V_1, Bottom: V_2

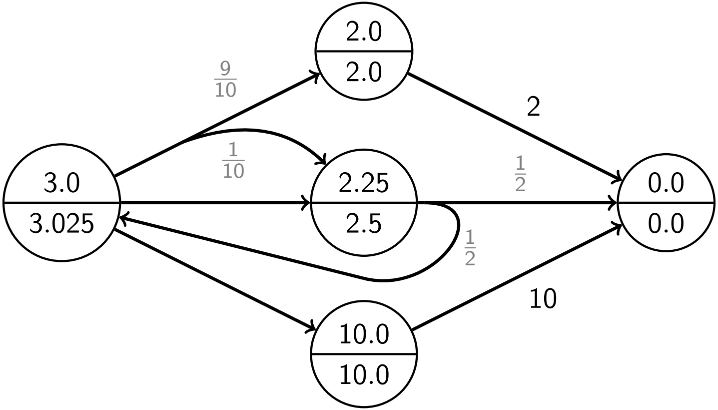


Top: V_2, Bottom: V_3



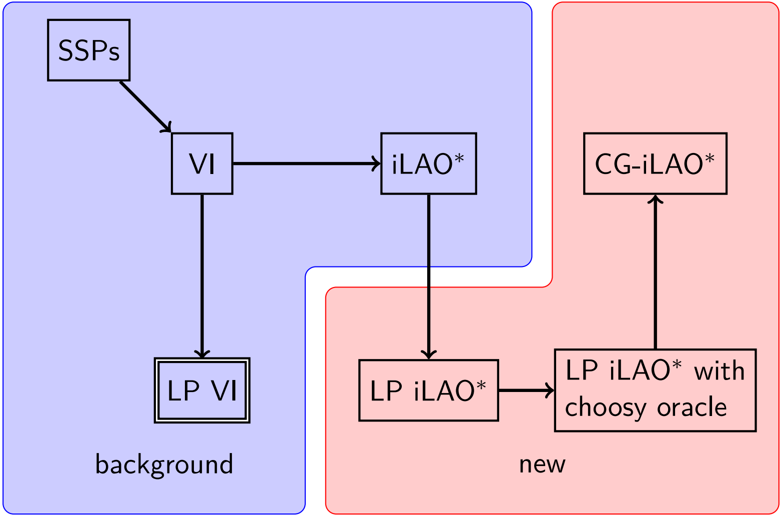
Top: V_3, Bottom: V_4

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au



Biggest change: $2.5 - 2.25 = 0.25$ — **we'll stop here.**

Top: V_4, Bottom: V_5


SSPs
VI
iLAO*
CG-iLAO*
LP VI
LP iLAO*
LP iLAO* with choosy oracle
background
new

VI LP

$$\max_{\vec{\mathcal{V}}} \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0 \qquad \forall g \in \mathrm{G}$$

$$\mathcal{V}_s \leq C(a) + \sum_{\text{states } s'} P(s'|s,a) \cdot \mathcal{V}_{s'} \qquad \forall s \notin \mathrm{G}, a \in \mathrm{A}(s)$$

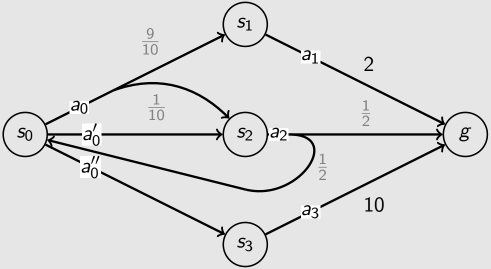
Reminder: Bellman Equations

$$V(g) = 0 \qquad \forall \text{ goals } g \in \mathrm{G}$$

$$V(s) = \min_{\text{actions } a} C(a) + \sum_{\text{states } s'} P(s'|s,a) \cdot V(s') \qquad \forall \text{ states } s \notin \mathrm{G}$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Example SSP



LP for our example

$$\max_{\vec{\mathcal{V}}} \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

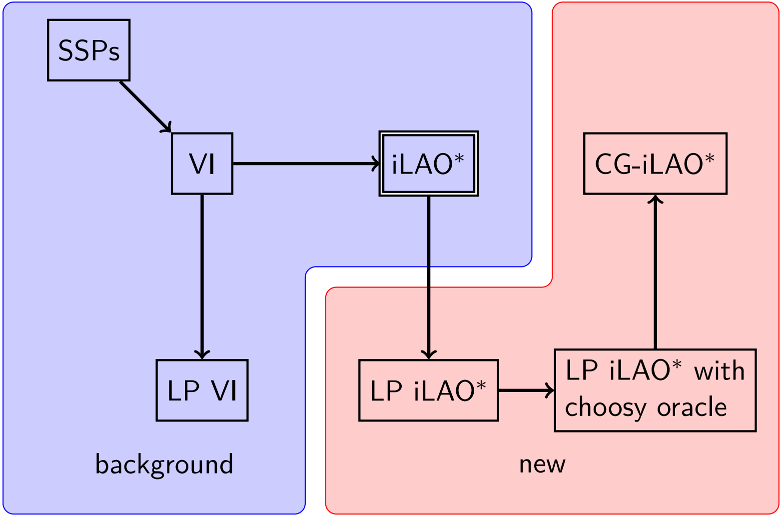
$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

$$\mathcal{V}_{s_1} \leq 2 + \mathcal{V}_g$$

$$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$$

$$\mathcal{V}_{s_3} \leq 10 + \mathcal{V}_g$$



SSPs

VI

iLAO*

CG-iLAO*

LP VI

LP iLAO*

LP iLAO* with choosy oracle

background

new

Definition: Heuristic

- Estimate of cost-go-to
- $H : \mathsf{S} \to \mathbb{R}_{\geq 0}$
- *Admissible* when $H(s) \leq V^*(s) \quad \forall s \in \mathsf{S}$

Terminology Warning: heuristic is just the name of the function; the algorithms are optimal

Blind vs Heuristic Search

- VI is a blind search algorithm — considers all states
- can use heuristics to guide search and focus on interesting states

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Let's use heuristics to prune states: iLAO* Hansen and Zilberstein (2001)

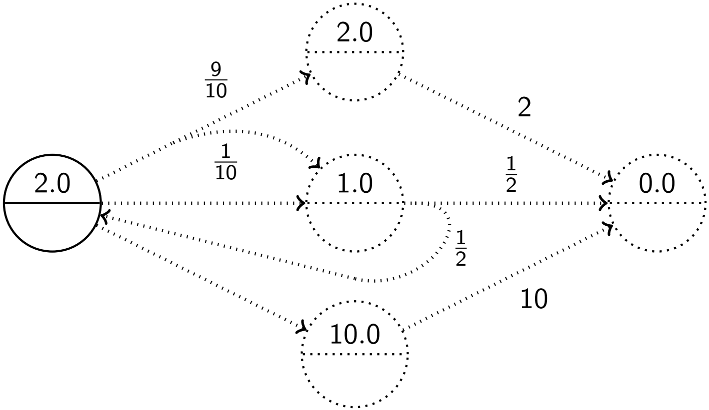
Key Idea 1: Solve partial SSPs

do not enumerate and solve whole reachable state space

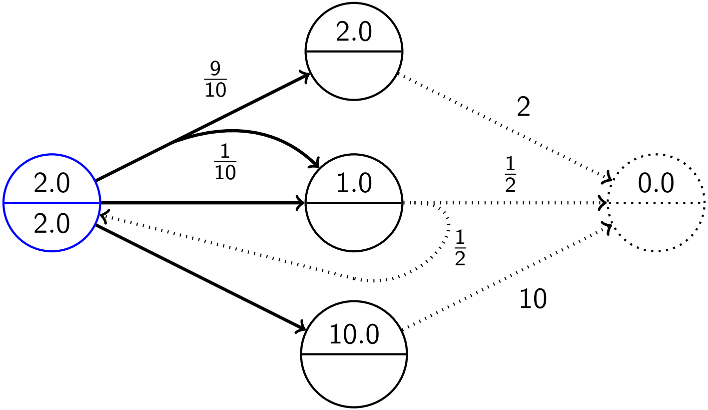
- look at partial SSP
- find best policy in partial SSP
- expand fringes using heuristic
- solve policy envelope
- repeat

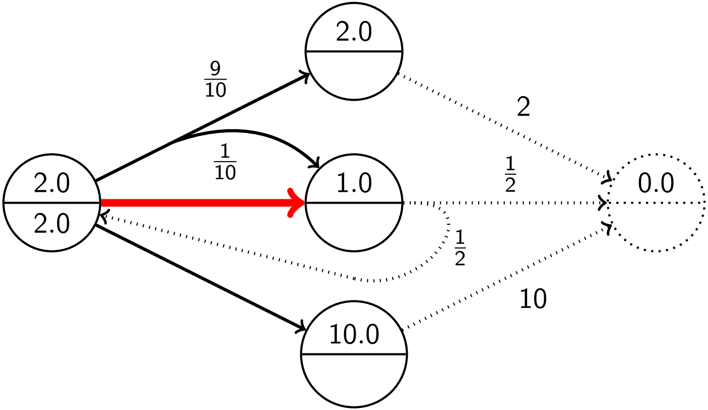
Key Idea 2: Don't solve fully

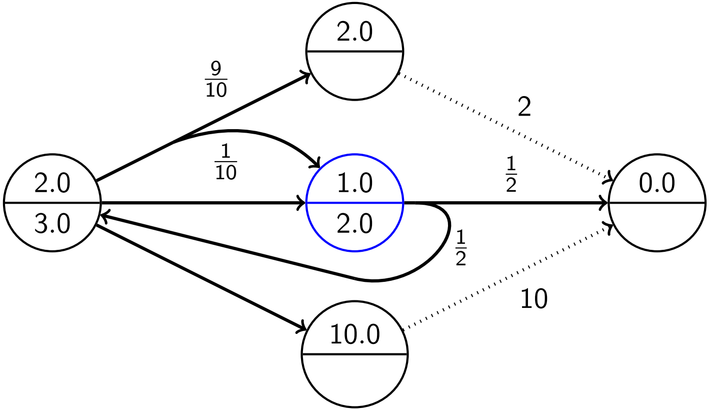
approximate V^* for intermediate partial SSPs with single pass of Bellman backups



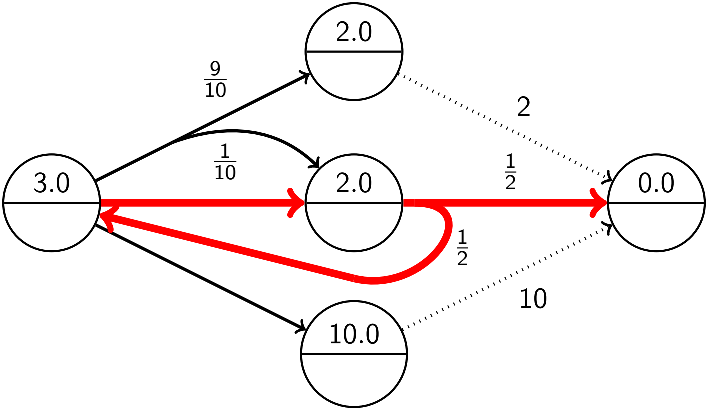
h is all-outcomes determinisation

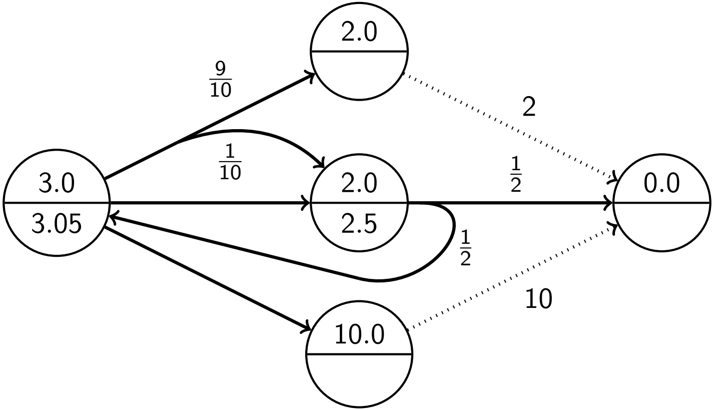

2.0
2.0
2.0
1.0
10.0
0.0
$\frac{9}{10}$
$\frac{1}{10}$
2
$\frac{1}{2}$
$\frac{1}{2}$
10

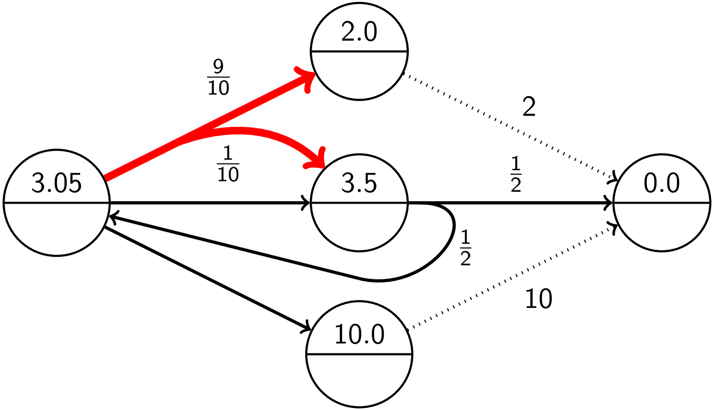

2.0
9/10
1/10
2.0
2.0
1.0
10.0
2
1/2
1/2
10
0.0



johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au


2.0
9/10
2
1/10
3.0
2.0
1/2
0.0
1/2
10
10.0


2.0
$\frac{9}{10}$
2
$\frac{1}{10}$
3.0
3.05
2.0
2.5
$\frac{1}{2}$
0.0
$\frac{1}{2}$
10
10.0

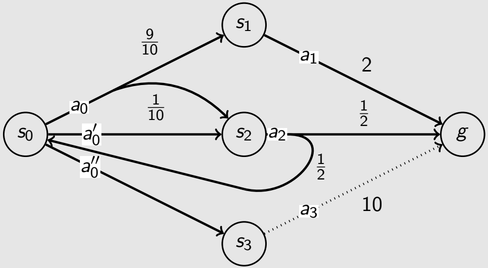


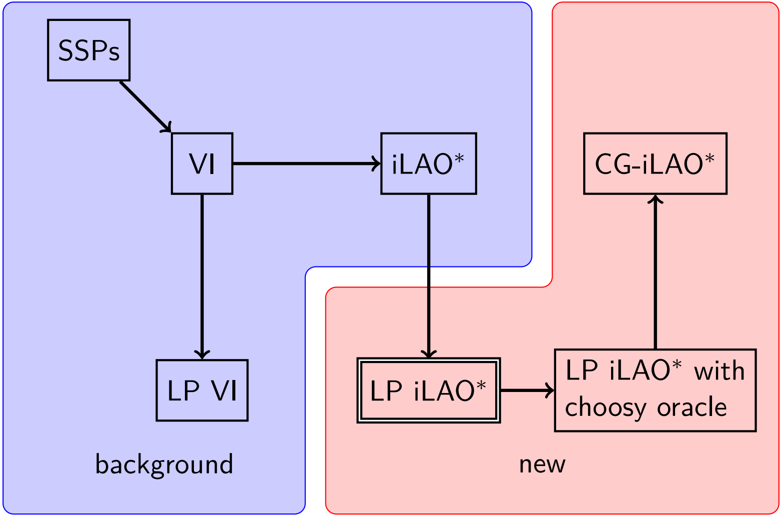
johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Inactive Actions

- s_3 is pruned by h
- Q-value for a_0'' is computed in **each** backup on s_0
- doing useless computation in each iteration of iLAO*!

Final Partial SSP





SSPs

VI

iLAO*

LP VI

CG-iLAO*

LP iLAO*

LP iLAO* with choosy oracle

background

new

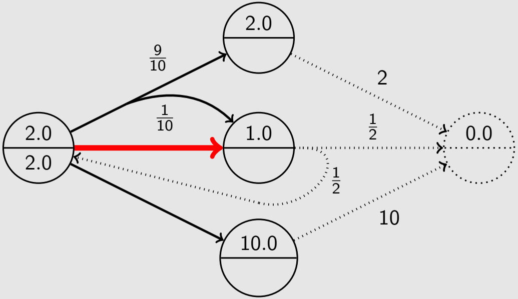
johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

iLAO* as Linear Program

- each partial SSP is a small linear program
- add states to partial SSP ☞ add variables to the LP
- add actions to partial SSP ☞ add constraints to the LP

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Partial SSP with Candidate Policy (Iter 1)



LP for our example

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\mathcal{V}_{s_2} = h(s_2) = 1$$

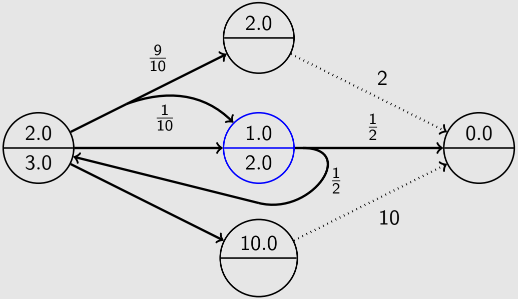
$$\mathcal{V}_{s_3} = h(s_3) = 10$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

Partial SSP with s_2 Expanded (Iter 2)



LP for our example

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\cancel{\mathcal{V}_{s_2} = h(s_2) = 1}$$

$$\mathcal{V}_{s_3} = h(s_3) = 10$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

$$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Variable Generation Desrosiers and Lübbecke (2005)

1. solve LP with subset of variables (RMP)
2. find variables to add with pricing problem
3. repeat until no variables to add

Constraint Generation

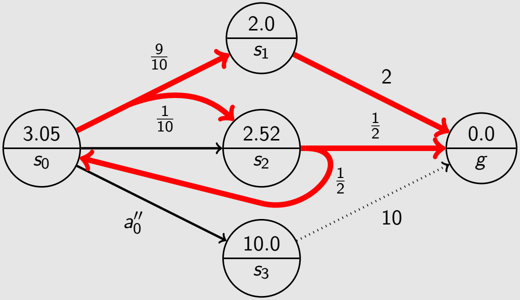
1. solve LP with subset of constraints (relaxed LP)
2. find constraints that are violated in original LP with *separation oracle*
3. repeat until no violated constraints

Separation Oracle

- INPUT: solution $\vec{x}$ to relaxed LP
- OUTPUT: if $\vec{x}$ violates constraint from original LP, return such constraint. Otherwise return nothing.
- iLAO*'s separation oracle: state expanded ⟶ constraints for applicable actions *may* be violated, so return all of them

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Final Partial SSP



Inactive Actions

- a_0'' is inactive
- constraint for (s_0, a_0'') is loose

LP for Final Partial SSP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\mathcal{V}_{s_3} = h(s_3) = 10$$

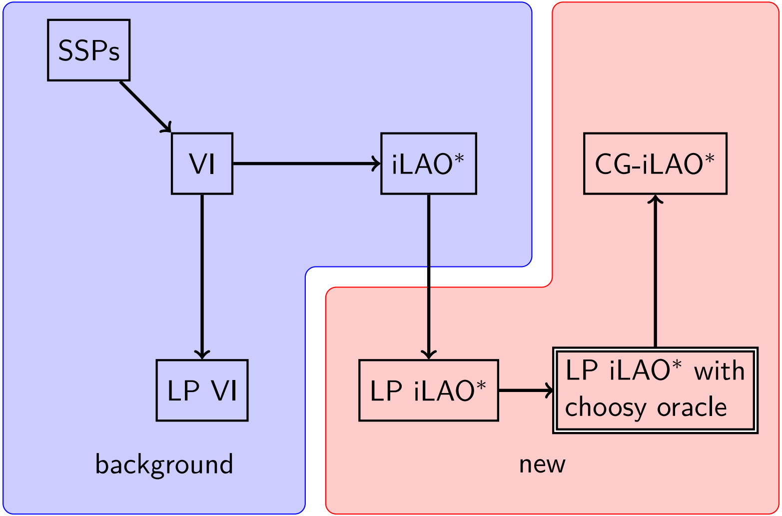
$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3} \quad ☜$$

$$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au



SSPs

VI

iLAO*

CG-iLAO*

LP VI

LP iLAO*

LP iLAO* with choosy oracle

background

new

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

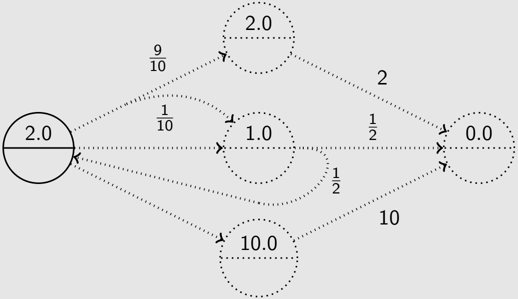
Choosy Oracle

- avoid loose constraints (which correspond to inactive actions)!
- only add constraints if actually violated

  ☞ ensures that the constraint will be active

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Partial SSP

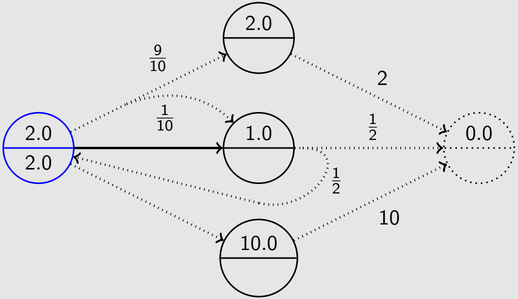


LP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$
$$\mathcal{V}_{s_0} = h(s_0) = 2$$

h is all-outcomes determinisation

Partial SSP



LP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\cancel{\mathcal{V}_{s_0} = h(s_0) = 2}$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\mathcal{V}_{s_2} = h(s_2) = 1$$

$$\mathcal{V}_{s_3} = h(s_3) = 10$$

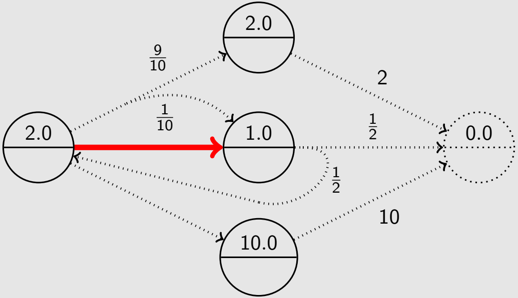
$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Partial SSP



LP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\mathcal{V}_{s_2} = h(s_2) = 1$$

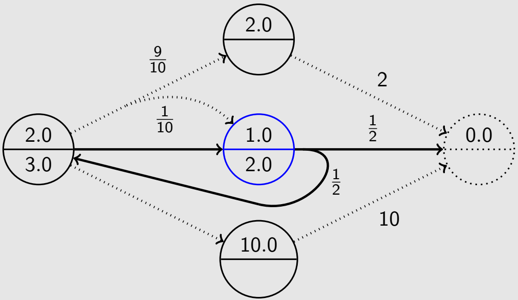
$$\mathcal{V}_{s_3} = h(s_3) = 10$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

Partial SSP



LP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\cancel{\mathcal{V}_{s_2} = h(s_2) = 1}$$

$$\mathcal{V}_{s_3} = h(s_3) = 10$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

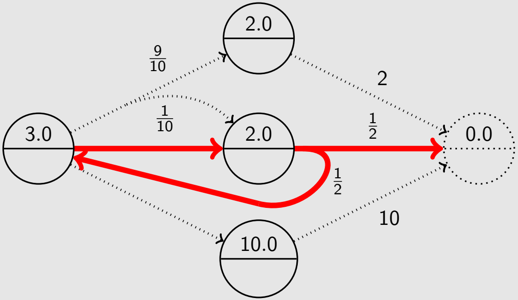
$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

$$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Partial SSP



LP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\mathcal{V}_{s_3} = h(s_3) = 10$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

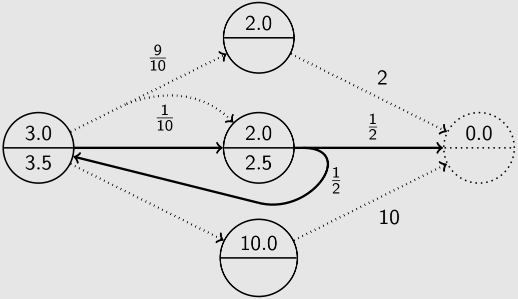
$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

$$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Partial SSP



LP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\mathcal{V}_{s_3} = h(s_3) = 10$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

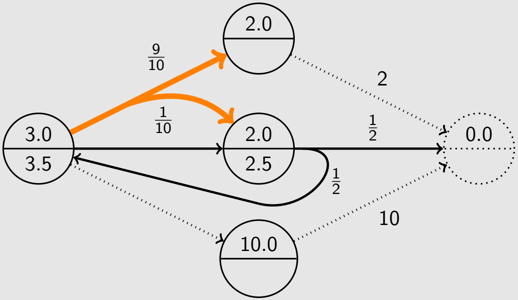
$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

$$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Partial SSP



LP

$$\max \mathcal{V}_{s_0} \text{ s.t.}$$

$$\mathcal{V}_g = 0$$

$$\mathcal{V}_{s_1} = h(s_1) = 2$$

$$\mathcal{V}_{s_3} = h(s_3) = 10$$

$$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$$

$$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$$

$$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$$

Note: (1) constraint violation on “old” action (2) $V(s_0) \not\leq V^*(s_0)$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

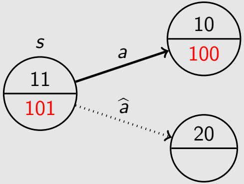
Naive Separation Oracle

```
for all states $s$ in partial SSP:
  for all actions $a \in A(s)$ outside partial SSP:
    check for violation on $(s, a)$, i.e., whether $Q(s, a) < V(s)$
```

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

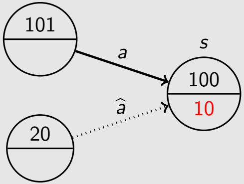
Efficient Separation Oracle

if $V(s)$ increases: check (s, a) for $a \in A(s)$
(don't need to check actions with constraints already added)



$11 \leq 1 + 10$ (a)✓
$11 \leq 1 + 20$ $(\hat{a})$✓
$101 \leq 1 + 100$ (a)✓
$101 \leq 1 + 20$ $(\hat{a})\times$

if $V(s)$ decreases: check (s', a') that lead to s



$101 \leq 1 + 100$ (a)✓
$20 \leq 1 + 100$ $(\hat{a})$✓
$101 \leq 1 + 10$ $(a)\times$
$20 \leq 1 + 10$ $(\hat{a})\times$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Fixing Violations

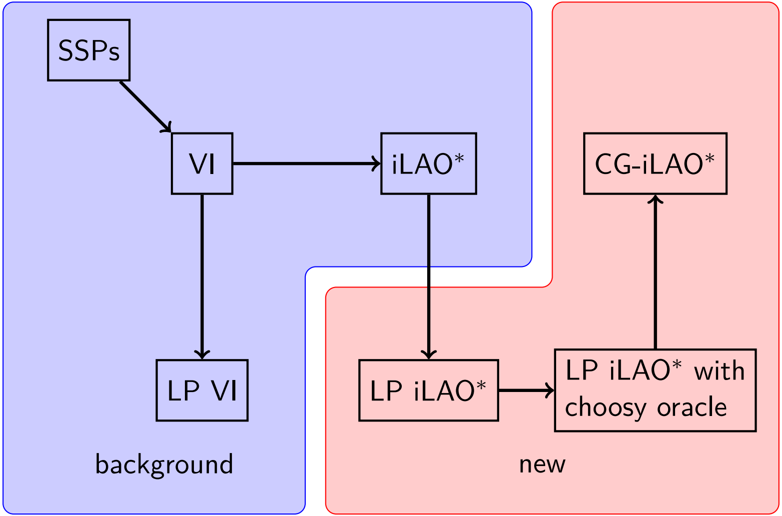
if $V(s) \not\leq Q(s, a)$:
  set $V(s) \leftarrow Q(s, a)$

☞ careful: may trigger another violation!

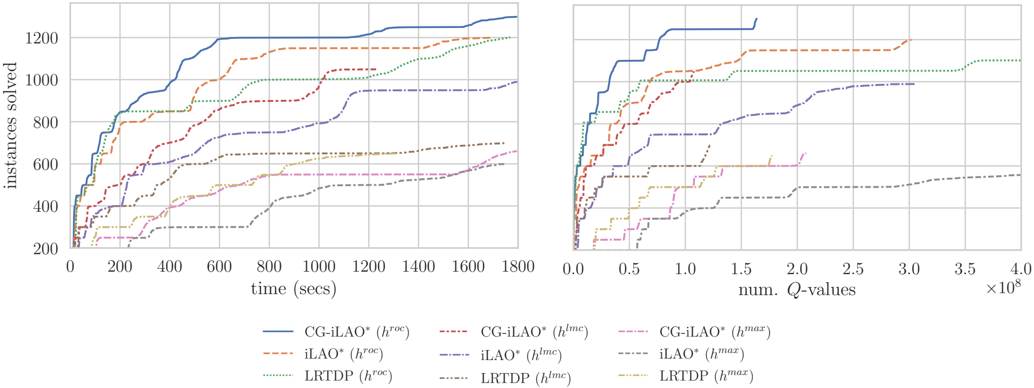
CG-iLAO*

CG-iLAO* is the dynamic programming implementation of LP iLAO* with choosy oracle!

$$\begin{aligned} \text{iLAO}^* &\longleftrightarrow \text{LP iLAO}^* \\ \text{CG-iLAO}^* &\longleftrightarrow \text{LP iLAO}^* \text{ with choosy oracle} \end{aligned}$$



johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

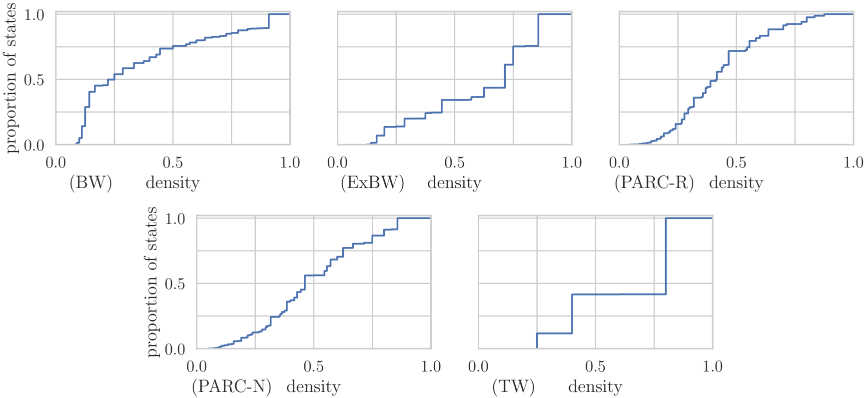


johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Coverage per Domain

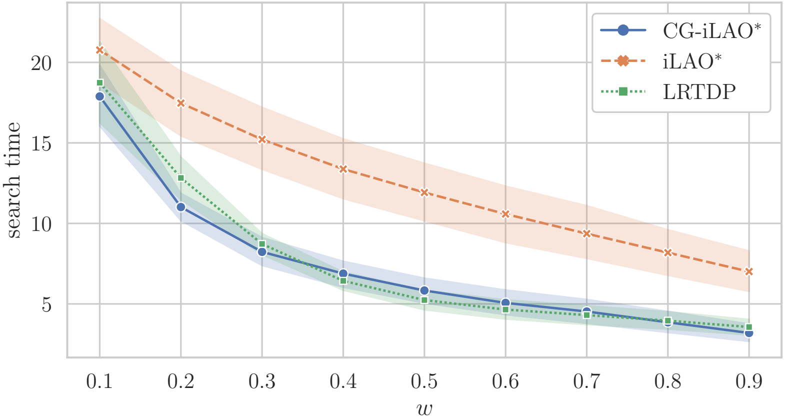
		BW	ExBW	PARC-N	PARC-R	TWH	**Total**
Num. of instances		300	250	300	250	200	1300
h^{roc}	CG-iLAO*	**300**	**250**	**300**	**250**	**200**	**1300**
	iLAO*	**300**	200	**300**	**250**	150	1200
	LRTDP	257	**250**	**300**	200	195	1202
h^{lmc}	CG-iLAO*	150	**250**	**300**	200	150	1030
	iLAO*	150	200	**300**	200	140	990
	LRTDP	0	200	**300**	50	149	699
$h^{\max}$	CG-iLAO*	150	200	150	0	161	661
	iLAO*	150	150	150	0	150	600
	LRTDP	150	200	150	0	150	650

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au



CG-iLAO* added 43–65% of iLAO*'s actions.

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au



Using artificial heuristic $h_w^{\text{pert}}(s) := V^*(s) \cdot$ uniform random value from $(w, 1]$

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Termination Condition: VI

- in the presence of cycles, VI only converges to V^* in the limit
- stop when Bellman error $\leq \epsilon$
- Bellman error is $\max_s |V_{i+1}(s) - V_i(s)|$

Termination Condition: iLAO*

- stop when ϵ-consistent AND no fringes
- ϵ-consistent when $\max_{s \in S^\pi} |V_{i+1}(s) - V_i(s)| \leq \epsilon$
- fringe states are artificial goals in the partial SSP that are not goals in the original

Termination Condition: CG-iLAO*

- stop when ϵ-consistent AND no fringes AND no constraint violations
- note: constraint violations are tracked with residual in our pseudocode

Intuition for iLAO*'s correctness

- invariant: $V \leq V^*$
- eventually no fringes remain
- eventually Bellman residual on greedy policy is $\leq \epsilon$

Intuition for CG-iLAO*'s correctness

Hand Waving: ϵ-consistency is straightforward, can think of variable generation and constraint generation.

Tricky bit: $V \nleq V^*$ so how can we make sure CG-iLAO* doesn't ignore states or actions we need?
☞ if $V(s)$ could decrease, it is tracked ☞ may be updated, then residual is tracked

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Summary

- iLAO* can be formulated in terms of linear programming
- iLAO* uses heuristic to prune states, but can't prune actions
  - ☞ wastes time on inactive actions!
- we refine iLAO*'s separation oracle
  - ☞ gives us CG-iLAO*
  - ☞ able to ignore inactive actions!
- CG-iLAO* beats the state-of-the-art!

Related Work

Action elimination Bertsekas (1995) can detect and prune useless actions, but requires upper bound; our approach (1) does not need an upper bound (2) adds actions as needed instead of pruning unecessary actions

More information available at `schmlz.github.io/cgilao`

johannes.schmalz@anu.edu.au, felipe.trevizan@anu.edu.au

Bellman, R. 1957. *Dynamic programming*. Princeton University Press.

Bertsekas, D. 1995. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific.

Desrosiers, J.; and Lübbecke, M. 2005. *A Primer in Column Generation*, 1–32. Springer US.

Geisser, F.; Poveda, G.; Trevizan, F.; Bondouy, M.; Teichteil-Königsbuch, F.; and Thiébaux, S. 2020. Optimal and Heuristic Approaches for Constrained Flight Planning under Weather Uncertainty. In *Proc. of 30th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Hansen, E.; and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*.