Efficient Constraint Generation for Stochastic Shortest Path Problems

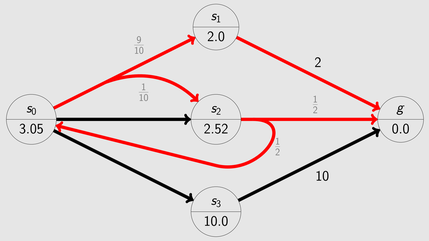
Johannes Schmalz and Felipe Trevizan

Australian National University

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

Stochastic Shortest Path Problems (SSPs)

SSPs are given by the tuple $\langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C \rangle$, where S, s_0, G, and C, as in classical planning problems, denote the states, initial state, goal states, actions, and cost function. $P(s'|s,a)$ gives the probability of reaching s' after applying a to s. This SSP is our running example:



V^* is shown in the bottom of each node. V^* is the optimal cost-to-go, and the unique function that satisfies the *Bellman equations*:

$$V(g) = 0 \qquad \forall \text{ goals } g \in \mathsf{G}$$
$$V(s) = \min_{\text{actions } a} \underbrace{C(a) + \sum_{\text{states } s'} P(s'|s,a) \cdot V(s')}_{Q(s,a)} \qquad \forall \text{ states } s \notin \mathsf{G}$$

The red actions denote the optimal policy. The greedy policy w.r.t. V^* is optimal.

Value Iteration (VI)

Value iteration starts with V_0 and iteratively applies *Bellman backups* until convergence:

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

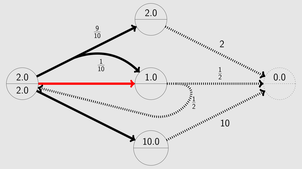
This Linear Program (LP) finds V^* and is called the VI LP:

$$\max_{\mathcal{V}} \mathcal{V}_{s_0} \text{ s.t.}$$
$$\mathcal{V}_g = 0 \qquad \forall \text{ goals } g \in \mathsf{G}$$
$$\mathcal{V}_s \leq C(a) + \sum_{\text{states } s'} P(s'|s,a) \cdot \mathcal{V}_{s'} \qquad \forall \text{ states } s \notin \mathsf{G},\ \text{applicable actions } a \in \mathsf{A}(s)$$

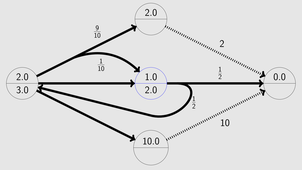
iLAO* (Hansen and Zilberstein 2001)

Key idea: use heuristic to guide search. At each step consider a *partial SSP*, a smaller SSP that we partially solve with a single pass of Bellman backups. The partial SSP is selected using V and the heuristic. Each partial SSP has its own VI LP.

Partial SSP (after 1 iteration)



Expand s_2



$\max \mathcal{V}_{s_0}$ s.t.
$\mathcal{V}_g = 0$
$\mathcal{V}_{s_1} = h(s_1) = 2$
~~$\mathcal{V}_{s_2} = h(s_2) = 1$~~
$\mathcal{V}_{s_3} = h(s_3) = 10$
$\mathcal{V}_{s_0} \leq 1 + \frac{9}{10}\mathcal{V}_{s_1} + \frac{1}{10}\mathcal{V}_{s_2}$
$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_2}$
$\mathcal{V}_{s_0} \leq 1 + \mathcal{V}_{s_3}$
$\mathcal{V}_{s_2} \leq 1 + \frac{1}{2}\mathcal{V}_{s_0} + \frac{1}{2}\mathcal{V}_g$

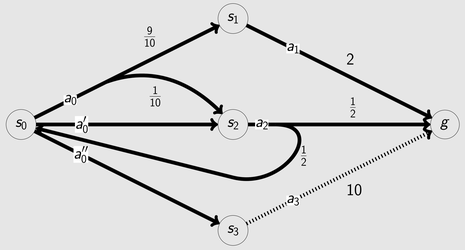
Inactive Actions

Action $a \in \mathsf{A}(s)$ is inactive if $V(s) < Q(s,a)$ with two interpretations:
1. Bellman backup to s does not use a
2. a's constraint in VI LP is loose

iLAO* Considers Inactive Actions

When iLAO* expands s, it adds all actions A(s). Some of these are not needed.

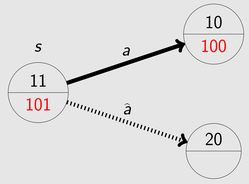
In this case iLAO* prunes s_3, but the action a_0'' is still considered and evaluated for each backup of s_0.



New Algorithm: CG-iLAO*— Ignores Inactive Actions

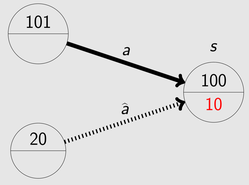
Key idea: when expanding a state, do not add inactive actions to the partial SSP.
When CG-iLAO* expands state s, it adds only the actions $\operatorname{argmin}_{a \in \mathsf{A}(s)} Q(s,a)$
Challenge: inactive actions may become active as V changes, how can we detect this?
Naive: check each $a \in \mathsf{A}(s)$ outside the partial SSP.
Better: only need to consider the following cases:

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



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

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



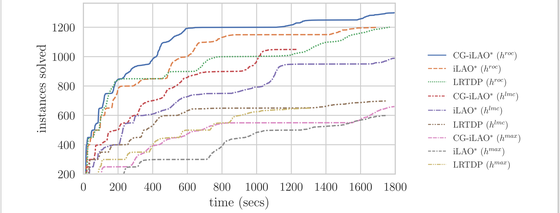
$101 \leq 1 + 100$ $(a)\checkmark$
$20 \leq 1 + 100$ $(\bar{a})\checkmark$
$101 \leq 1 + 10$ $(a)\times$
$20 \leq 1 + 10$ $(\bar{a})\times$

Performance Results

We compare CG-iLAO* with iLAO* (Hansen and Zilberstein 2001) and LRTDP (Bonet and Geffner 2003) using the heuristics $h^{\max}$, h^{lmc} (Helmert and Domshlak 2009), h^{roc} (Trevizan, Thiébaux, Haslum 2017).

CG-iLAO* has the best coverage!

Cumulative Plot of Instances Solved w.r.t. Time

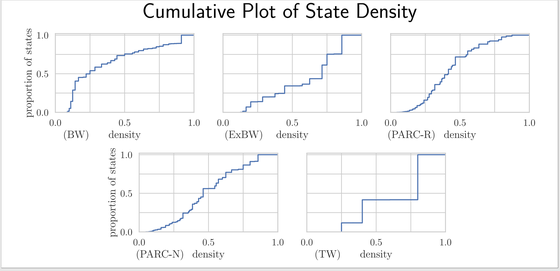


Coverage per Domain

		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

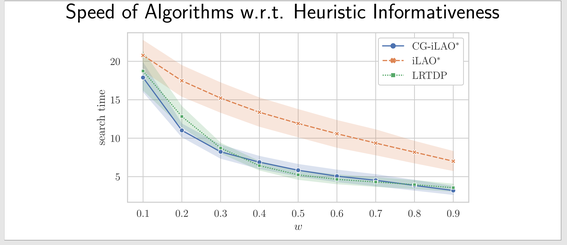
Understanding the Improvement

Cumulative Plot of State Density



There are many states where CG-iLAO* with h^{roc} adds small proportion of available actions; CG-iLAO* added 43–65% of iLAO*'s actions.

Speed of Algorithms w.r.t. Heuristic Informativeness



Using $h^{\text{pert}}_w(s) := V^*(s) \cdot$ uniform random value from $(w, 1]$. As heuristic gets more informative (w increases) the gap between CG-iLAO* and LRTDP increases w.r.t. iLAO*. CG-iLAO* scales better with informative heuristic.