Finding Optimal Deterministic Policies for Constrained Stochastic Shortest Path Problems

Johannes Schmalz and Felipe Trevizan

Background

Constrained Stochastic Shortest Path Problem (CSSP)

Defined by tuple $\langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, \vec{C}, \vec{u} \rangle$ with states (S), initial state (s_0), goals (G), actions (A), transition probabilities (P), cost vector ($\vec{C} : \mathsf{S} \to \mathbb{R}^{n+1}_{\geq 0}$), cost upper bounds ($\vec{u} \in \mathbb{R}^n_{\geq 0}$).

Policies

Policies π map states to actions. They have two flavours:

1. **Deterministic** policy $\pi : \mathsf{S} \to \mathsf{A}$
2. **Stochastic** policy $\pi : \mathsf{S} \to$ probability distribution over A

Policy Cost: $C_i(\pi) =$ expected cost i to reach G with π

Feasibility: π must satisfy $C_i(\pi) \leq u_i$ for all $i \in \{1, \ldots, n\}$

Optimality: π is optimal if it is feasible and minimises $C_0(\pi)$

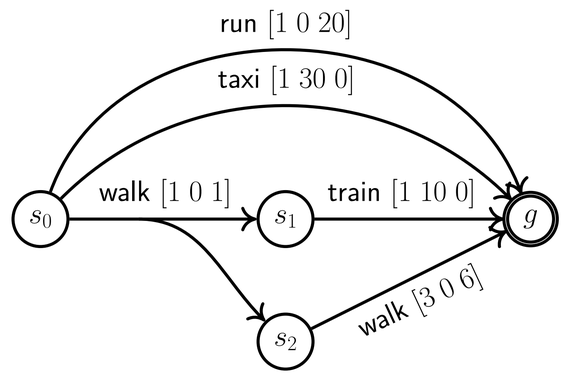
Optimal stochastic policies are optimal for the CSSP, but optimal deterministic policies need **not** be optimal for the CSSP. Nevertheless...

Practitioners Want Deterministic Policies

- ethical issues in medical contexts (Roijers et al. 2013)
- aviation regulations (Geißer et al. 2020)
- coordination in multi-agent systems (Dolgov and Durfee 2005)
- accountability and explainability (Hong and Williams 2023)
- more predictable (see example)

CSSP Example

You need to get to work. You can run, use a taxi, or walk to the train station and try the train. The train is cancelled with 50% probability. Each action has a cost vector $[t\ p\ e]$ in terms of time (t), price (p), and personal effort (e). **Task:** get to work in minimal time s.t. price ≤ 15 and effort ≤ 10 over expectation.



Optimal stochastic policy: $\pi^*(s_0) = \{\text{run} : 50\%, \text{taxi} : 50\%\}$
$C_{\text{time}}(\pi^*) = 1$ $C_{\text{price}}(\pi^*) = 15$ $C_{\text{effort}}(\pi^*) = 10$
But individually *run* and *taxi* violate the constraints...

Optimal deterministic policy: $\pi(s) =$ walk or train
$C_{\text{time}}(\pi) = 3$ $C_{\text{price}}(\pi) = 5$ $C_{\text{effort}}(\pi) = 4$
More expensive time-wise, but satisfies constraints*

Solving CSSPs with MIP

Imagine the CSSP as a flow network: actions are pipes, states are junctions, and we want to route 1L of water through it with minimal cost. Each $x_{s,a}$ denotes amount of flow through pipe a from s and $\Delta_{s,a} = 1/0$ if the pipe has/has no flow.

$$
\begin{aligned}
&\min_{\vec{x},\vec{\Delta}} C_0(\vec{x}) \text{ s.t.} && \\
&out(s) - in(s) = 0 && \forall s \in \mathsf{S} \setminus (\mathsf{G} \cup \{s_0\}) \\
&out(s_0) - in(s_0) = 1 && \\
&\sum_{g \in \mathsf{G}} in(g) = 1 && \\
&x_{s,a} \geq 0 && \forall s \in \mathsf{S}, a \in \mathsf{A}(s) \\
&C_i(\vec{x}) \leq u_i && \forall i \in \{1, \ldots, n\} \\
&x_{s,a} \leq \mathbf{M}\Delta_{s,a} && \forall s \in \mathsf{S}, a \in \mathsf{A}(s) \\
&\sum_{a \in \mathsf{A}(s)} \Delta_{s,a} \leq 1 && \forall s \in \mathsf{S} \\
&\Delta_{s,a} \in \{0, 1\} && \forall s \in \mathsf{S}, a \in \mathsf{A}(s)
\end{aligned}
$$

In and out flow from states, and flow cost are macros:

- $C_i(\vec{x}) = \sum_{s \in \mathsf{S}, a \in \mathsf{A}(s)} x_{s,a} C_i(a)$
- $out(s) = \sum_{a \in \mathsf{A}(s)} x_{s,a}$
- $in(s) = \sum_{s' \in \mathsf{S}, a' \in \mathsf{A}(s')} x_{s,a} P(s'|s,a)$

Contributions

The Issue with big $\mathbf{M}$

If $\mathbf{M}$ is too small...

- MIP can become infeasible
- optimal policy may become infeasible

If $\mathbf{M}$ is too big...

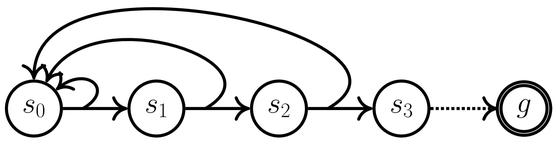
- Numerical instability
- May get non-integer solutions (trickle flow)

A General Bound

This always works: $\mathbf{M} = p_{\min}^{-|\mathsf{S} \setminus \mathsf{G}|}$

- $p_{\min} \in (0, 1]$ min. probability in CSSP
- S \ G set of non-goal states

That's impractically big, but can't do better in general:



Finding big $\mathbf{M}$ Automatically

New insight: we can relate **any feasible** solution $\vec{x}$ to the maximum flow over the **optimal** solution with $\text{obj}(\vec{x}) \cdot \underline{g}^{-1} \geq x_{\max}$ where $\underline{g} = \min_{a \in \mathsf{A}} C_0(a)$.

Algorithm:

1. select some $\mathbf{M}$
2. try to solve MIP with $\mathbf{M}$
   - if infeasible: increase $\mathbf{M}$ and repeat step 2
   - if feasible: set $\mathbf{M} \leftarrow \text{obj}(\vec{x}) \cdot \underline{g}^{-1}$
3. solve MIP with $\mathbf{M}$

Avoiding big $\mathbf{M}$

Can completely avoid big $\mathbf{M}$ with SOS1 constraints. A SOS1 constraint is an ordered set of continuous variables $\{x_0, \ldots, x_k\}$ such that at most one variable is allowed to be nonzero.

For us: $\{x_{s,a} | a \in \mathsf{A}(s)\}$ for each state s, i.e., at most one action for $\mathsf{A}(s)$ may have nonzero flow. That's exactly what we want!

New Algorithm for Finding Deterministic Policies

For finding **stochastic** policies, i^2-dual (Trevizan et al. 2017) is the state-of-the-art. It uses a heuristic to iteratively construct partial CSSPs, focusing on the promising states. The partial CSSPs are solved with LPs.

New: we replace i^2-dual's LPs with MIPs. This yields i^2-dual-det, which finds optimal **deterministic** policies.

Making it faster: we don't care about the exact solution to each MIP, so we can approximate them! There are many ways to do this:

- Use LP relaxation
- Use constraint generation for integrality constraints
- Allow large MIP gap (also makes it anytime)

More Background: Linearisation

A **linearisation** $\vec{\lambda} \in \mathbb{R}^{n+1}$ relaxes the CSSP into an SSP with the scalar cost function $C'(a) = \vec{\lambda} \cdot \vec{C}(a)$

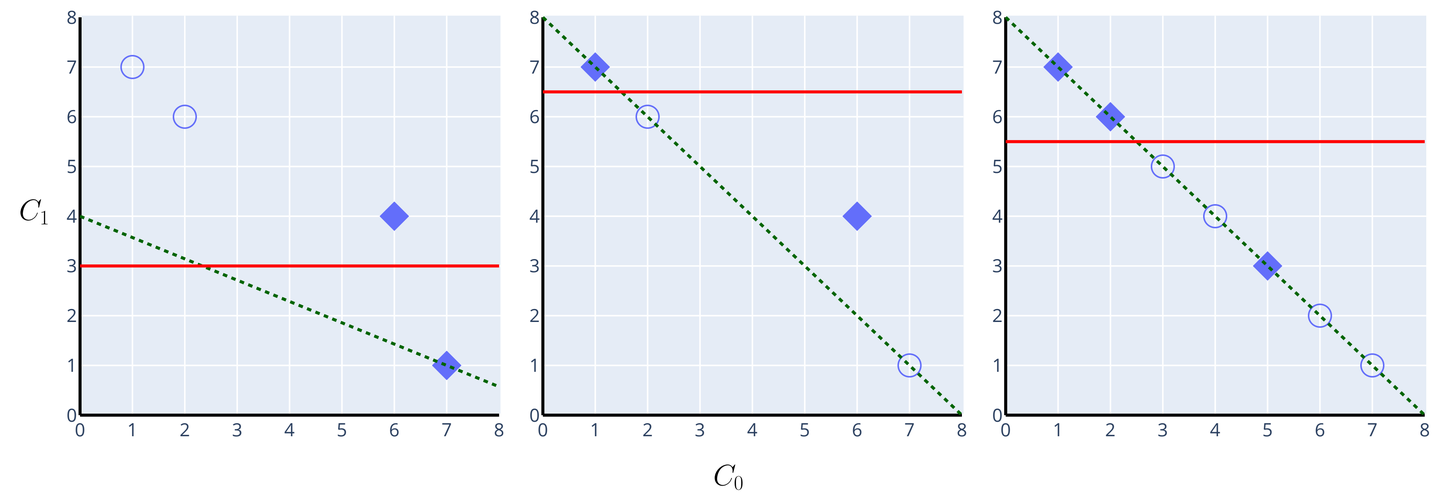
$\vec{\lambda} = [1\ 0\ \cdots\ 0]$ is the trivial relaxation that only looks at C_0.

Constraint Interestingess

We introduce a **new categorisation of CSSPs** based on how interesting their constraints are.

- **Trivial**: if the constraints can be ignored, and the trivial relaxation yields a feasible policy.
- **Linearisable**: if $\exists \vec{\lambda}$ so that its relaxation yields a feasible policy.
- **Interesting**: if it is not linearisable.
- Practically Interesting: if it is linearisable, but for each feasible policy there are many infeasible ones.

Constraint Interestingess Visualised



Policy costs w.r.t. C_0 and C_1. Red line is constraint over C_1. Dotted green line is contour line for linearisation. Points are the costs of deterministic policies. ♦/○ = feasible / infeasible w.r.t. C_2

Very Short Summary of Experimental Results

- The existing state-of-the-art, which is based on linearisation, is best on **trivial** and **linearisable** problems.
- Our algorithm i^2-dual-det is best on **interesting problems**.

More available at `schmlz.github.io/det-pi-for-cssp`

You can email me at `johannes.schmalz@anu.edu.au`