Finding Optimal Deterministic Policies for Constrained Stochastic Shortest Path Problems

Johannes Schmalz and Felipe Trevizan

Australian National University
ORCID (Johannes Schmalz): https://orcid.org/0000-0003-3389-2490, ORCID (Felipe Trevizan): https://orcid.org/0000-0001-5095-7132

Abstract. Constrained Stochastic Shortest Path problems (CSSPs) are a modelling framework for probabilistic problems with a primary cost and constraints over secondary costs such as fuel consumption or monetary budget. While the optimal solution for a CSSP is usually a stochastic policy, practical considerations often demand deterministic solutions, for instance, in aviation and multi-agent systems. Previous works have addressed this issue for special cases of CSSPs; in this work, we show the technical issues in generalising these results and show how they can be addressed. Then, using these methods, we extend the state-of-the-art heuristic search method for finding optimal stochastic policies to efficiently find deterministic policies for CSSPs. We show experimentally that our algorithm competes with the state-of-the-art, and is able to solve the class of problems with difficult-to-satisfy constraints on which the state-of-the-art fails.

1 Introduction

Constrained Stochastic Shortest Path problems (CSSPs) model planning problems with probabilistic action effects where the expected cost of actions is minimised while satisfying resource constraints, for instance, plan a flight route that minimises the expected fuel usage while arriving in a given time window and limiting the time spent in bad weather zones [11]. Solutions to CSSPs are given by policies, functions that map a state that the agent may encounter into a decision. Policies can be deterministic, where the decision for a state is a single action; or stochastic, where the agent must randomly select an action from a probability distribution. In contrast to its unconstrained counterpart, there is no guarantee that any optimal policy for CSSPs will be deterministic. Moreover, for finding optimal stochastic CSSP policies there are efficient polynomial-time algorithms, whereas the best deterministic policies need not be optimal and finding them is NP-complete [9]. Nevertheless, there are practical scenarios in which deterministic policies are preferred or required: in multi-agent systems it is difficult to coordinate between agents with stochastic policies [8]; making probabilistic decisions in medical contexts can have ethical issues [16]; aviation regulations may not permit stochastic policies [11]; and more generally, there can be issues with accountability and explainability [14]. To further motivate why deterministic policies may be preferred to stochastic ones, consider the following toy problem, which is over-simplified (e.g. it only has deterministic actions) but reflects issues encountered in practice (e.g. [21]). The task is to move from A to B and three move actions are available with different monetary (M), fuel (F), and time (T) costs: *slow* (M: $1, F$: $1, T$: 7), *medium* (M: $7, F$: $5, T$: 4), and *fast* (M: $5, F$: $9, T$: 2). We want a policy that minimises M under the constraints $F \leq 5$ and $T \leq 5$. The optimal stochastic policy is *move slow with probability* 0.6 *and move fast with probability* 0.4. Its expected costs are $M = 2.6, F = 4.2, T = 5$, but individually, *slow* and *fast* violate the time and fuel constraints respectively. In many practical applications, the deterministic policy *move medium with probability 1* is preferred, even if its monetary cost is higher.[1]

Dolgov and Durfee [8] developed a Mixed Integer Program (MIP) that finds the cheapest deterministic policy for Constrained Markov Decision Processes (CMDPs). CSSPs and CMDPs are similar, but it is not trivial to adapt this MIP to CSSPs. In this paper we explain why, and develop the first domain-independent methods to make this adaptation. Recently, Hong and Williams [14] have proposed a two-stage heuristic-search algorithm for finding deterministic policies for CSSPs, which we call LAny. Its first stage solves a relaxation of the CSSP and the second stage uses this relaxed solution to find an optimal deterministic policy. LAny is effective on many problems, but fails when the relaxation is a poor approximation of the CSSP, which happens when the secondary costs interact in a way that makes the constraints difficult to satisfy simultaneously. To understand this failure, we introduce a categorisation of CSSPs based on how difficult their constraints are to satisfy: in ascending order of difficulty, CSSPs can be trivial, linearisable, or interesting. To overcome the shortcoming of the state-of-the-art on interesting CSSPs, we introduce i^2-dual-det, which adapts i^2-dual [22], the state-of-the-art heuristic search algorithm for finding optimal stochastic policies to the context of finding the cheapest deterministic policies. Our experiments show that i^2-dual-det dominates all other planners on interesting CSSPs.

The structure of this paper is as follows: in secs. 2 and 3 we give the background for CSSPs, CMDPs, and the LP for finding optimal stochastic policies. In secs. 4 and 5, we present and discuss Dolgov and Durfee [8]'s MIP, adapted in a canonical way to find optimal deterministic policies for CSSPs. We explain that this adaptation is incomplete, and our first contribution is to address this. In sec. 6, we explain i^2-dual and present our second contribution: the new algorithm i^2-dual-det and techniques to make i^2-dual-det more efficient. In sec. 7 we explain LAny [14]. In sec. 8, we theoretically compare LAny and i^2-dual-det, and define a way to categorise CSSPs into trivial, linearisable, and interesting problems, as our third contribution. Our experimental results and methodology are in sec. 9.

[1] An execution of a deterministic policy might still violate one of the constraints due to stochastic actions and the guarantee is over expectation only.

2 Constrained Stochastic Shortest Path Problems

Stochastic Shortest Path problems (SSPs) [4] are given by the tuple $\mathbb{S} = \langle \mathsf{S}, s_I, \mathsf{G}, \mathsf{A}, P, C \rangle$ where S is the finite set of states; $s_I \in \mathsf{S}$ is the initial state; $\mathsf{G} \subset \mathsf{S}$ is the set of goal states; A is the finite set of actions and $\mathsf{A}(s)$ denotes the actions applicable in state s; $P(s'|s,a)$ gives the probability of reaching s' after applying a to s; $C : \mathsf{A} \to \mathbb{R}_{>0}$ is the cost function where $C(a)$ denotes the cost of applying a.

A deterministic policy π is a partial function $\pi : \mathsf{S} \to \mathsf{A}$, which tells the agent to apply action $\pi(s)$ when in state s. Stochastic policies $\pi : \mathsf{S} \to \text{dist}(\mathsf{A})$ are more general: given a state, they return a probability distribution over actions, from which the agent should pick an action randomly; $\pi(s,a)$ denotes the probability with which a should be applied in s. A policy π is closed on $\mathbb{S}$ if π is defined for all states that can be reached by following π from s_I on $\mathbb{S}$. A policy π is proper if following π from s_I reaches G from s_I with probability 1; π is improper otherwise. Let $\phi^\pi_s(t)$ denote the probability of being in state s after following π from s_I for t steps. Then, the expected cost of π, written $C(\pi)$, is defined as $C(\pi) \coloneqq \sum_{t=0}^{\infty} \sum_{s \in \mathsf{S}, a \in \mathsf{A}} \phi^\pi_s(t)\pi(s,a)C(a)$.

We make the following standard assumptions for SSPs: (1) *improper policies have positive infinite cost*; and (2) *from any $s \in \mathsf{S}$, there exists a sequence of actions that reach a goal with probability greater than 0.* The latter is known as the reachability assumption and it implies the existence of a proper policy and ensures that all non-goal states have at least one applicable action. An optimal policy for SSPs is any policy with minimal expected cost. For SSPs, at least one of its optimal policies is deterministic [4].

Constrained SSPs [1] extend SSPs by adding secondary costs for each action and imposing constraints over these. Formally, a CSSP is the tuple $\mathbb{C} = \langle \mathsf{S}, s_I, \mathsf{G}, \mathsf{A}, P, \vec{C}, \vec{u} \rangle$ where $\vec{C}$ is a cost vector $[C_0, \ldots, C_n]$ with strictly positive primary cost $C_0 : \mathsf{A} \to \mathbb{R}_{>0}$ and non-negative secondary costs $C_i : \mathsf{A} \to \mathbb{R}_{\geq 0}$ for $i \in \{1, \ldots, n\}$; and $\vec{u}$ is a vector of upper bounds $[u_1, \ldots, u_n]$ that induce the secondary cost constraints, with $u_i \in \mathbb{R}_{>0}$ for $i \in \{1, \ldots, n\}$. All other terms of the tuple are identical to SSPs. By ignoring the secondary costs and constraints we get the SSP relaxation of $\mathbb{C}$, $\langle \mathsf{S}, s_I, \mathsf{G}, \mathsf{A}, P, C_0 \rangle$. The expected cost of a policy π w.r.t. cost function C_i is written $C_i(\pi)$. A policy π is feasible for the CSSP only if $C_i(\pi) \leq u_i$ for all $i \in \{1, \ldots, n\}$. For CSSPs, optimal policies π must be feasible and minimise the expected primary cost $C_0(\pi)$. As a consequence of adding secondary cost constraints, it is not guaranteed that among the optimal solutions for a CSSP there are any deterministic ones [1, 20]. As discussed before, there may be reasons to require deterministic policies, so we call a feasible deterministic policy that minimises $C_0(\pi)$ an optimal deterministic policy.

Infinite-horizon Markov Decision Processes (MDPs), as used by Dolgov and Durfee [8], are the tuple $\mathbb{M} = \langle \mathsf{S}, \mathsf{A}, P, R \rangle$ where $\mathsf{S}, \mathsf{A}, P$ are defined identically to SSPs and $R : \mathsf{A} \to \mathbb{R}$ is a reward function, replacing the SSP's cost function. MDPs have no goal states, and instead of an initial state, we must specify an initial condition $\alpha \in \text{dist}(\mathsf{S})$, where $\alpha(s) \in [0,1]$ can be interpreted as the probability of starting in s. Additionally, we must specify a discount factor $\gamma \in [0,1)$. Analogously to SSPs, let $\phi^\pi_s(t)$ be the probability of being in state s after following π from α in t steps. For MDPs, optimal policies π maximise the total expected discounted reward criterion $U(\pi, \alpha, \gamma) \coloneqq \sum_{t=0}^{\infty} \sum_{s \in \mathsf{S}, a \in \mathsf{A}} \gamma^t \phi^\pi_s(t)\pi(s,a)R(a)$.

Constrained MDPs (CMDPs), again as used by Dolgov and Durfee [8], introduce $\vec{u} \in \mathbb{R}^n_{>0}$ as for CSSPs, and replace the reward function with $\vec{C} = [C_0, \ldots, C_n]$ with $C_i : \mathsf{A} \to \mathbb{R}$, where C_0 is still interpreted as a reward function, but $C_1, \ldots, C_n$ are secondary costs. For cost $i \in \{1, \ldots, n\}$, π's total expected discounted cost is given by $C_i(\pi, \alpha, \gamma) \coloneqq \sum_{t=0}^{\infty} \sum_{s \in \mathsf{S}, a \in \mathsf{A}} \gamma^t \phi^\pi_s(t)\pi(s,a)C_i(a)$. Policy π is feasible iff it satisfies $C_i(\pi, \alpha, \gamma) \leq u_i$ for each $i \in \{1, \ldots, n\}$. As with SSPs and CSSPs, MDPs always have optimal policies that are deterministic, whereas CMDPs may only have strictly stochastic optimal policies.

To motivate that adapting methods for (C)MDPs to (C)SSPs is non-trivial, we cite the result that infinite-horizon MDPs are a special case of SSPs [3]. The intuition behind this result is that at each step in infinite-horizon MDPs the agent "stops" with probability $1 - \gamma$, i.e., the accumulation of rewards stops. This is equivalent to every action reaching the goal of an SSP with probability $1 - \gamma$ [3]. Therefore, all closed policies for MDPs are proper and have finite cost/reward, in contrast to SSPs, where closed and improper policies do exist.

3 Finding Stochastic Policies for CSSPs

The fundamental algorithm for finding optimal stochastic policies for CSSPs is the Linear Program (LP) over occupation measure shown in LP 1 [7, 20]. Here, $x_{s,a}$ is the occupation measure of (s,a), $[\cdot]$ are Iverson brackets, and $\mathit{in}(s)$ and $\mathit{out}(s)$ are macros that encode the actions leading into and exiting s respectively. A solution $\vec{x}$ induces the stochastic policy $\pi(s,a) = x_{s,a}/\mathit{out}(s)$. The value of $x_{s,a}$ can be read as the expected number of times we encounter s and then apply a when following the induced policy from s_I.

LP 1 can be interpreted as a network flow problem, where a unit of flow is injected into s_I, and must be routed through a system of pipes (representing actions) such that all flow reaches the goal states. $\mathit{in}(s)$ represents the amount of flow that enters s from all actions, weighted by the probability that they reach s; and $\mathit{out}(s)$ is the amount of flow leaving s through its outgoing actions. The network respects preservation of flow (C1), i.e., the amount of flow entering a state must be equal to the flow exiting it. The exceptions are s_I where a unit of flow is injected, and the goal states which are sinks. To ensure a proper policy, one unit of flow must enter the sink (C2). Each $x_{s,a}$ represents the expected number of times we encounter s and then apply a when following the induced policy from s_I.

$$\min_{\vec{x}} \sum_{s \in \mathsf{S}, a \in \mathsf{A}(s)} x_{s,a} \cdot C_0(a) \text{ s.t. C1–C6} \qquad \text{(LP 1)}$$

$$\mathit{out}(s) - \mathit{in}(s) = [s = s_I] \qquad \forall s \in \mathsf{S} \setminus \mathsf{G} \quad \text{(C1)}$$

$$\sum_{s_g \in \mathsf{G}} \mathit{in}(s_g) = 1 \qquad \text{(C2)}$$

$$\mathit{out}(s) = \sum_{a \in \mathsf{A}(s)} x_{s,a} \qquad \forall s \in \mathsf{S} \quad \text{(C3)}$$

$$\mathit{in}(s) = \sum_{s' \in \mathsf{S}, a' \in \mathsf{A}(s')} x_{s',a'} \cdot P(s|s',a') \qquad \forall s \in \mathsf{S} \quad \text{(C4)}$$

$$x_{s,a} \geq 0 \qquad \forall s \in \mathsf{S}, a \in \mathsf{A}(s) \quad \text{(C5)}$$

$$\sum_{s \in \mathsf{S}, a \in \mathsf{A}(s)} x_{s,a} \cdot C_j(a) \leq u_j \qquad \forall j \in \{1, \ldots, n\} \quad \text{(C6)}$$

4 Finding Deterministic CSSP Policies with MIPs

Dolgov and Durfee [8] present a Mixed Integer Program (MIP) for finding deterministic policies for CMDPs, which can be adapted to CSSPs [14], as shown in MIP 1. This MIP extends LP 1 to find optimal deterministic policies. The binary variable $\Delta_{s,a}$ for each $s \in \mathsf{S}, a \in \mathsf{A}(s)$ is an indicator for whether $x_{s,a} > 0$ by C9; C7 ensures that zero or one distinct actions are applied in each state, i.e.,

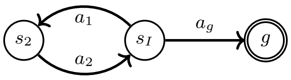

Figure 1. A CSSP where an improper policy exists and Dolgov and Durfee [8]'s LP for finding M is unbounded.

that the policy is deterministic. M is a constant value in $\mathbb{R}_{>0}$ which must be large enough so that $x_{s,a} \leq M$, otherwise the MIP may disallow the optimal solution or become infeasible.

$$\min_{\vec{x},\vec{\Delta}} \sum_{s\in\mathsf{S},a\in\mathsf{A}(s)} x_{s,a}\cdot C_0(a) \text{ s.t. C1–C9} \tag{MIP 1}$$

$$\sum_{a\in\mathsf{A}(s)} \Delta_{s,a} \leq 1 \tag{C7}$$

$$x_{s,a} \leq M\cdot\Delta_{s,a} \qquad \forall s\in\mathsf{S}, a\in\mathsf{A}(s) \tag{C8}$$

$$\Delta_{s,a}\in\{0,1\} \qquad \forall s\in\mathsf{S}, a\in\mathsf{A}(s) \tag{C9}$$

So far, adapting the MIPs to CSSPs has been straightforward, but now we encounter the issue of selecting M. If M is too small the optimal solution may become infeasible, or the whole MIP may become infeasible. This presents a dangerous situation: if the user is not certain that M is large enough, and the MIP is infeasible, they can not be sure if it is because the model is really infeasible or M is too small. If M is excessively large it can impact performance, introduce numerical instability, and may allow integrality constraints to be violated (called trickle flow), yielding non-deterministic policies.

Often M is chosen manually by a user with domain knowledge, but it is very difficult to find such upper bounds on large problems with complex interactions and cycles. For CMDPs, Dolgov and Durfee [8] present an LP whose optimal objective gives a valid M. Generalising to CSSPs, this LP would be $\max_{\vec{x}} \sum_{s\in\mathsf{S},a\in\mathsf{A}(s)} x_{s,a}$ s.t. C1–C5. However, for CSSPs this LP may be unbounded — this is where the techniques for CMDPs fail for CSSPs. As an example of this phenomenon, consider the CSSP in fig. 1 with deterministic actions, unit costs, and no secondary cost constraints. The solution $x_{s_I,a_g}=1$ and $x_{s_I,a_1}=x_{s_2,a_2}=y$ satisfies the LP for any $y\in\mathbb{R}_{\geq 0}$, which can be made arbitrarily large to maximise the objective, i.e., the LP is unbounded. This problem occurs when a closed improper policy exists in the CSSP and this explains why the issue does not exist for CMDPs: all policies for CMDPs are proper, thanks to the discount factor.

5 Finding M for CSSPs

In this section, we first derive new theoretical results to bound M for CSSPs. Then, using the new theory, we show practical ways to derive M without domain knowledge.

For a given CSSP $\mathbb{C}$, we define a lower bound on positive probabilities $p_{\min} := \min\{P(s'|s,a) : s,s'\in\mathsf{S}, a\in\mathsf{A}, P(s'|s,a)>0\}$ and bounds on primary action costs $\overline{g} := \max_{a\in\mathsf{A}} C_0(a)$ and $\underline{g} := \min_a C_0(a)$. For a feasible solution $\vec{x}$ of MIP 1 which induces a proper feasible policy for $\mathbb{C}$, we define $x_{\max} := \max_{s\in\mathsf{S},a\in\mathsf{A}} x_{s,a}$.

Recall that M is valid as long as $M\geq x^*_{\max}$ for $\vec{x}^*$ associated with π^*. We start by giving a domain-independent formula for deriving an upper bound on $x_{\max}$ over all deterministic policies:

Theorem 1. *For any $\mathbb{C}$ and deterministic proper policy encoded by $\vec{x}$, $p_{\min}^{-|\mathsf{S}\setminus\mathsf{G}|}$ is an upper bound for $x_{\max}$. Moreover, $\mathbb{C}$ and $\vec{x}$ exist such that the bound is tight.*

Proof. To show this is an upper bound consider $\vec{x}$ that induces the deterministic proper policy π; and for contradiction, suppose for some $s\in\mathsf{S}$ and $a\in\mathsf{A}(s)$ we have $x_{s,a}=x_{\max}=(1+\epsilon)\cdot p_{\min}^{-|\mathsf{S}\setminus\mathsf{G}|}$ where $\epsilon\in\mathbb{R}_{>0}$. To derive a contradiction, we show that the amount of flow that reaches goals from s, a exceeds 1. Since π is proper, there exists an acyclic trajectory $\langle s_1,\ldots,s_k\rangle$ with $s_1=s$ and $s_k\in\mathsf{G}$. Observe that for $i\in\{1,\ldots,k-1\}$ we have $in(s_{i+1})\geq out(s_i)\cdot P(s_{i+1}|s_i,\pi(s_i))$ and $out(s_i)\geq in(s_i)$. We collapse these inequalities into $in(s_k)\geq out(s_1)\cdot\prod_{i=1}^{k-1}P(s_{i+1}|s_i,\pi(s_i))\geq out(s_1)\cdot p_{\min}^{k-1}$. Recall that $s_1=s$ and $x_{s,a}=x_{\max}$, so $out(s_1)\cdot p_{\min}^{k-1}=x_{\max}\cdot p_{\min}^{k-1}=(1+\epsilon)\cdot p_{\min}^{k-1-|\mathsf{S}\setminus\mathsf{G}|}$. But, $k-1\leq|\mathsf{S}\setminus\mathsf{G}|$ since an acyclic trajectory can pass through at most $|\mathsf{S}\setminus\mathsf{G}|$ states before encountering a goal. Therefore $(1+\epsilon)\cdot p_{\min}^{k-1-|\mathsf{S}\setminus\mathsf{G}|}\geq 1+\epsilon>1$ which violates C2, yielding the desired contradiction.

Now, we show the bound is tight by giving $\mathbb{C}$ and $\vec{x}$ where $x_{\max}=p_{\min}^{-|\mathsf{S}\setminus\mathsf{G}|}$. We give a CSSP with a single cost function, i.e., an SSP. The backwards-cycling chain SSP has $\mathsf{S}=\{s_0,\ldots,s_k\}$; $s_0=s_I$; $\mathsf{G}=\{s_k\}$; $\mathsf{A}(s_i)=\{a_i\}$ for $0\leq i<k$; $P(s_{i+1}|s_i,a_i)=p_{\min}$ and $P(s_0|s_i,a_i)=1-p_{\min}$; and C is not important. This SSP has a single solution $\vec{x}$ which satisfies $x_{s_0,a_0}=p_{\min}^{-|\mathsf{S}\setminus\mathsf{G}|}$ for any $k\in\mathbb{N}_{>0}$, which can be shown by straightforward induction proof. □

Thm. 1 gives an easy way to compute valid M for any CSSP, but, for a large state-space this bound is impractically large, so we turn our attention to practical bounds. First, we enumerate some bounds on $x_{\max}$ that work for deterministic policies in special cases:

1. If $\mathbb{C}$ is acyclic, then $x_{\max}=1$.
2. Consider the case where each action incurs some secondary cost, i.e., $\sum_{i=1}^{n}C_i(a)>0\ \forall a\in\mathsf{A}$, and there are constraints associated with each of these costs. Then, we can construct the LP $\max_{\vec{x}}\sum_{s\in\mathsf{S},a\in\mathsf{A}(s)}x_{s,a}$ s.t. C1–C6 that includes the secondary cost constraints. This LP is bounded: if there were some $x_{s,a}$ that tended to infinity, then there must be i for which $\sum_{s\in\mathsf{S},a\in\mathsf{A}}x_{s,a}C_i(a)$ tends to infinity and violates its constraint. The solution will yield an upper bound on $x_{\max}$.
3. If we have an upper bound U for the primary cost, we observe that MIP 1 has the same optimal solution under the constraint
$$\textstyle\sum_{s\in\mathsf{S},a\in\mathsf{A}(s)} x_{s,a}\cdot C_0(a)\leq U \tag{C10}$$
For this new MIP, the LP $\max_{\vec{x}}\sum_{s\in\mathsf{S},a\in\mathsf{A}(s)}x_{s,a}$ s.t. C1–C5, C10 will give an upper bound on $x_{\max}$, and the LP must be bounded because primary action costs are strictly positive and C10 forces the LP's primary cost to be bounded. Subsequently, this is a valid choice of M for MIP 1, with the important caveat that some solutions whose objectives exceed U may be infeasible under this M. In this sense, a smaller value of M can have positive impact on the solving time by pruning suboptimal solutions.
4. Following on from the previous point, we observe that $U\geq\sum_{s\in\mathsf{S},a\in\mathsf{A}(s)}x^*_{s,a}\cdot C_0(a)\geq\sum_{s\in\mathsf{S},a\in\mathsf{A}(s)}x^*_{s,a}\cdot\underline{g}=\underline{g}\sum_{s\in\mathsf{S},a\in\mathsf{A}(s)}x^*_{s,a}$, where $\vec{x}^*$ is the optimal solution. This lets us relate U, the upper bound on cost, to the flow of $\vec{x}^*$ thus: $U\cdot\underline{g}^{-1}\geq\sum_{s\in\mathsf{S},a\in\mathsf{A}(s)}x^*_{s,a}\geq x^*_{\max}$. Note that action costs must be strictly positive for this bound to be finite.

We now exploit these insights to develop alg. 1, which solves the MIP and automatically finds a valid M. The idea is that we initialise M with some value $M_0\in\mathbb{R}_{>0}$ that may be too small, and then we increase M exponentially until we find a feasible solution for the MIP. The objective of this feasible solution, denoted obj$(\vec{x})$, is an upper bound U on the primary cost of the original MIP with arbitrary

Algorithm 1: Find M automatically

```
Function solve(MIP $\mathcal{L}, M \in \mathbb{R}_{>0}, \kappa \in \mathbb{R}_{>1}$)
    for $i \in \{0, 1, \dots\}$
        $\vec{x} \leftarrow$ optimise $\mathcal{L}$ with $M = \kappa^i \cdot M_0$
    until $\vec{x}$ is feasible
    if $obj(\vec{x}) \cdot \underline{g}^{-1} > M$ then
        $\vec{x} \leftarrow$ optimise $\mathcal{L}$ with $M = \text{obj}(\vec{x}) \cdot \underline{g}^{-1}$
    return $\vec{x}$
```

M, so we can leverage special case 4: update $M \leftarrow U \cdot \underline{g}^{-1}$, and re-solve the MIP, now yielding an optimal policy. Importantly, this algorithm only works if the input CSSP $\mathbb{C}$ has a feasible solution; if $\mathbb{C}$ is infeasible, alg. 1 will never terminate.

To conclude this section, we show a way to avoid M altogether. At the core of finding a deterministic policy, we want to ensure that for each s, $x_{s,a} > 0$ for at most one $a \in \mathsf{A}(s)$. Operations Research has a mechanism for encoding this: an ordered set of variables $\{x_0, \dots, x_k\}$ can be added to an LP or MIP as Special Ordered Sets of type 1 (SOS1), which enforces that x_i may be non-zero for at most one i [2]. These constraints are enforced with a branch-and-bound algorithm. So, for forcing deterministic policies, instead of adding binary indicator variables and tying them to LP 1, we can add to LP 1 the ordered sets $\{x_{s,a} : a \in \mathsf{A}(s)\}$ for each $s \in \mathsf{S}$ as SOS1 constraints. Note that the order of states can be chosen arbitrarily. SOS1 constraints are a standard feature in LP and MIP solving software, e.g., Gurobi and CPLEX support them.

For the remainder of this paper, we refer to LP 1 modified to yield deterministic policies a MIP, regardless whether it is implemented with one of the big-M approaches or with SOS1 constraints.

6 Efficiently Computing Deterministic Policies

In this section, we introduce *i^2-dual-det*, our new algorithm for finding optimal deterministic policies for CSSPs. To do this, we first present i^2-dual [22], the state-of-the-art heuristic search algorithm for computing optimal stochastic policies for CSSPs, then combine it with our results from sec. 5, yielding our novel algorithm.

A key shortcoming of using LP 1 to find optimal stochastic policies is that it does not scale to large problems, as the LP requires a variable per state-action pair and a constraint per state. I^2-dual addresses this issue by using a heuristic function $H : \mathsf{S} \rightarrow \mathbb{R}$ to estimate the cost of reaching the goal from different states and focus on the subset of the search space that appears least expensive. I^2-dual generalises A* to CSSPs and its high-level pseudocode is shown in alg. 2.

At each step, i^2-dual has a partial CSSP $\widehat{\mathbb{C}}$, which is a copy of $\mathbb{C}$ with a subset of states as normal; the remaining non-goal states s are called fringes and are treated as artificial goals with a single-time terminal cost vector of $\vec{T}(s)$, determined by a vector of admissible heuristics. In each iteration of the inner loop, i^2-dual checks all the reachable states under its candidate policy π, called the envelope of π, and expands any fringe states in the policy by adding them to $\mathbb{C}$ as normal states. Then, i^2-dual solves LP 1 for the new $\widehat{\mathbb{C}}$, and updates π accordingly. Importantly, the candidate policy will avoid fringe states that are deemed expensive by the heuristics, enabling i^2-dual to ignore unpromising parts of the state-space.

The other defining feature of i^2-dual is that, instead of calling an external heuristic to determine the termination costs of partial CSSPs, the LP for computing its heuristic is embedded within the LP for solving these partial CSSPs. I^2-dual uses as its heuristic $h^{\text{c-pom}}$ [22] which considers multiple simplifications of the original CSSP that preserve some of the original problem's structure, called atomic projections. Each projection is a CSSP that can be solved with LP 1. $h^{\text{c-pom}}$ builds LP 1 for each projection and ties these LPs together by forcing each action to be applied the same number of times in expectation for each projection. Then, to solve each partial CSSP $\widehat{\mathbb{C}}$, i^2-dual merges LP 1 for $\widehat{\mathbb{C}}$ with the LP for $h^{\text{c-pom}}$ so that the search for the optimal policy and computation of $h^{\text{c-pom}}$ are intertwined.

Algorithm 2: i^2-dual

```
1  Function find-opt-stoch-policy(CSSP $\mathbb{C}$)
2      initialise partial CSSP $\widehat{\mathbb{C}}$ with fringe $s_I$
3      initialise empty candidate policy $\pi$
4      while envelope($\pi$) contains fringes do
5          $\widehat{\mathbb{C}} \leftarrow$ expand fringes
6          $\vec{x} \leftarrow$ solve LP 1 for $\widehat{\mathbb{C}}$ with $h^{\text{c-pom}}$
7          if infeasible then
8              return infeasible
9          $\pi \leftarrow$ policy from $\vec{x}$
10     return $\pi$
```

Now, we introduce i^2-dual-det, our novel heuristic search algorithm for finding deterministic policies for CSSPs. This is a straightforward modification of i^2-dual, where instead of using LP 1 to solve $\widehat{\mathbb{C}}$ in line 6, we use MIP 1 with SOS1 constraints or equipped with a method for deriving M automatically, as described in sec. 5. This MIP is integrated with the LP for $h^{\text{c-pom}}$. Importantly, the LP for $h^{\text{c-pom}}$ remains unchanged, preserving its admissibility for stochastic policies, hence also for deterministic policies. Thus, each π is a deterministic policy that must be cheaper than the optimal deterministic policy due to the fringes' admissibility. Once the policy envelope contains no fringes, it must be the optimal deterministic policy. The optimality of i^2-dual-det is inherited from i^2-dual, and can be proved with a simple modification of the proof for i-dual [20].

We conclude this section by discussing some performance improvements to i^2-dual-det. One important implementation detail for i^2-dual-det is that solving the MIP for the current $\widehat{\mathbb{C}}$ should use the solution from the previous iteration as a warm-start. This significantly speeds up search because modern MIP solvers can leverage such warm-starts to find a new solution faster, e.g., they can, in some cases, transform the old solution into a feasible solution for the new problem. Another technique for speeding up i^2-dual-det, is to solve relaxations of the intermediate partial CSSPs. An optimal solution for $\widehat{\mathbb{C}}$ is not necessary to expand useful fringes, and an approximate solution obtained from a relaxation can suffice. Of course, in the final steps, we must solve $\widehat{\mathbb{C}}$ optimally to ensure i^2-dual-det's optimality, but considerable time can be saved by approximating the intermediate partial CSSPs. Note that this is analogous to the improvement from LAO* to iLAO* [12]. We present two approximation techniques that use LP relaxations of the MIPs, and a third method that is compatible with the first two and is based on optimality gaps.

1. Solve LP relaxations of each MIP and when a closed policy is found, solve the MIPs. In other words, run i^2-dual first, then use its search tree and solution in i^2-dual-det. This was done by Hong and Williams [14] for i-dual [20].
2. Start by considering LP relaxations. Solve relaxations of each MIP and when a closed policy π is found determine at which states π is stochastic, then add determinising constraints only at these states. Repeat these steps until the policy is deterministic. This is

effectively constraint generation.

3. MIP solvers keep track of the optimality gap, which is the gap between its lower and upper bounds. Usually solvers run until the gap is close to zero in order to solve the problem optimally (up to a small error). In contrast, we relax our MIPs by allowing the solver to stop early with a larger optimality gap. Once i^2-dual-det finds a feasible solution with no fringe states but a large optimality gap, it has found a closed feasible policy for the CSSP. At this stage, we impose that the optimality gap must be close to zero, forcing future solutions to be optimal, and continue running i^2-dual-det with the feasible solution we found as a warm start.

I^2-dual-det can be implemented with item 1, item 2, or neither; and in parallel, any of these variants may use item 3. Note that item 3 lets i^2-dual-det behave like an anytime algorithm, since it can return closed feasible policies before termination, and the MIP's optimality gap directly gives an optimality gap for the policy. For maximum performance, the key is to find a balance between finding a solution quickly, obtaining a sufficiently informative solution so that useful fringes are expanded, and maximising how effectively the solver can reuse information from the previous iteration with warm starts.

7 Related work: LAny

In 2023, Hong and Williams [14] published an algorithm that we call the Lagrangian Anytime algorithm (LAny). It is an anytime heuristic-search planner for finding optimal deterministic policies. LAny works in two stages. Stage 1 finds the Lagrangian dual for the CSSP, which is effectively an approximation of the CSSP. Formally, MIP 1 can be re-expressed as $\min_{\pi\in\Pi} f(\pi)$ s.t. $g_i(\pi) \leq 0$ for $i \in \{1,\dots,n\}$ where Π is the set of deterministic policies, $f(\pi) = C_0(\pi)$ and $g_i(\pi) = C_i(\pi) - u_i$. Given a vector $\lambda \in \mathbb{R}^n_{\geq 0}$ with a non-negative entry for each secondary cost function, the CSSP's Lagrangian function is $L(\lambda,\pi) = f(\pi) + \lambda^T \cdot g(\pi)$. Then, a Lagrangian relaxation of the problem is $L(\lambda) = \min_{\pi\in\Pi} L(\lambda,\pi)$. In words, $L(\lambda)$ relaxes the constraints associated with each g_i, but penalises and rewards their violation or satisfaction by the linear term λ_i respectively. We call the SSP associated with $L(\lambda)$ the λ-SSP. For any $\lambda \in \mathbb{R}^n_{\geq 0}$, $L(\lambda)$ is a lower bound on the cost of $\pi^* \in \Pi$ for the original CSSP. Finally, the Lagrangian dual is the largest of these lower bounds, i.e., $L^* = L(\lambda^*) = \max_{\lambda\in\mathbb{R}^n_{\geq 0}} L(\lambda)$. For linear programs this dual has the same value as the optimal objective in the original problem (strong duality), but since we are restricting to deterministic policies we can only guarantee that λ^* is a lower bound on the cost of $\pi^* \in \Pi$ for the original CSSP (weak duality).

Finding λ^* is a piece-wise linear concave optimisation problem which LAny solves with an exact line search. This is an iterative approach that (1) starts with $\lambda = \vec{0}$, (2) solves λ-SSP, (3) computes a new λ that will improve the Lagrangian, and then loops back to (2) until it finds λ^*. LAny solves each λ-SSP with a version of iLAO* [12] that has been modified to use a vector of value functions $\vec{V} : \mathsf{S} \to \mathbb{R}^{n+1}_{\geq 0}$. The vector of costs and value functions are collapsed into a single value by taking their dot product with $\lambda' = [1, \lambda_0, \dots, \lambda_n]$. Since iLAO* is a heuristic-search algorithm, it benefits from initialising $\vec{V}$ with an admissible vector of heuristics $\vec{H}$. For $\vec{H}$ to be admissible, $\vec{H} \cdot \lambda'$ must be admissible for λ-SSP, which we can only guarantee in general by requiring that each H_i is admissible w.r.t. cost function i. This precludes heuristics such as $h^{\text{c-roc}}$ [22] that consider interactions between cost functions.

Throughout stage 1, LAny tracks the incumbent solution π_{inc}, i.e., the best feasible solution it has seen so far. Each time LAny solves a λ-SSP to get policy π, it checks whether π is feasible for $\mathbb{C}$ and has $C_0(\pi) < C_0(\pi_{\text{inc}})$, and if so, sets π as the new incumbent. The incumbent's primary cost gives an upper bound (ub) on the optimal policy cost. If no feasible solution has been found then there is no incumbent, and we define $C_0(\pi_{\text{inc}})$ and respectively ub to be ∞. At the same time, we get a lower bound (lb) because $L(\lambda) \leq C_0(\pi^*)$ for all λ, so LAny keeps as a lower bound $\max_\lambda L(\lambda)$ over all the λs it has considered so far. It can happen the optimality gap $ub - lb$ is smaller than some $\epsilon \in \mathbb{R}_{>0}$, which proves π_{inc} is an optimal solution (up to ϵ) and LAny can return it. It is also possible that there is no incumbent, that it is not optimal, or that it is optimal but LAny can not prove it is optimal because the lower bound is not tight. If stage 1 can not terminate by proving its incumbent is optimal, it keeps searching until it finds λ^*, and then enters stage 2.

Algorithm 3: Stage 2 of LAny

```
Function closing-gap($\mathbb{C}, \lambda^* \in \mathbb{R}^{n+1}_{\geq 0}, \pi_{inc}$)
    $lb \leftarrow L(\lambda^*)$
    for $i \in \{0, 1, \dots\}$
        $\pi \leftarrow i$th cheapest policy on $\lambda^*$-SSP
        $lb \leftarrow \max\{lb, L(\lambda^*, \pi)\}$
        if $\pi$ is $\mathbb{C}$-feasible and $C_0(\pi) < C_0(\pi_{inc})$ then
            $\pi_{\text{inc}} \leftarrow \pi$
    until $lb \geq C_0(\pi_{inc})$
    return $\pi_{\text{inc}}$
```

Stage 2 uses λ^* to find the optimal policy. As shown in alg. 3, stage 2 enumerates the policies on the λ^*-SSP in order of cost. Note that there may be many policies with equal cost; for selecting the ith cheapest policy we arbitrarily select one of the tied-cheapest policies, and the remaining policies of same cost are picked next. To find the ith cheapest policy, the algorithm creates a copy of $\mathbb{C}$ where some states' applicable actions are restricted so that previously seen, cheaper policies are disallowed, then uses the same variant of iLAO* as before to find the cheapest new policy. For more details and explanation for why this algorithm terminates optimally we defer to the original paper [14]. Note that compared to state-based search, stage 2 can be extremely inefficient as it searches over the policy space, which is considerably larger than the state space. The algorithm is only practical when the Lagrangian dual gives a good approximation of the CSSP, and only a small number of policies have to be enumerated. Hong and Williams [14] define an approximate variant of stage 2 that prunes unpromising policies. This can speed up search but sacrifices optimality and has no termination condition.

8 Theoretical Comparison

Optimal vs Anytime We define an algorithm to be optimal if it is guaranteed to eventually terminate with a proven optimal solution (up to ϵ); and anytime if it can produce feasible incumbent solutions and improves on them over time. i^2-dual-det and all its variants are optimal. By default, i^2-dual-det is not anytime, but variants that allow large optimality gaps for intermediate partial CSSPs, as per item 3, are anytime. LAny is both optimal and anytime.

Use of Heuristics Both LAny and i^2-dual-det are heuristic search algorithms, but they use them in different ways. LAny uses its variant of iLAO* in both stages. This iLAO* is supplied with a vector of heuristics which must be admissible per cost function, and therefore can not take into account the interaction between cost functions or the secondary cost constraints. These heuristics are computed by projecting the CSSP onto each of its cost functions C_i and ignoring the constraints, resulting in the SSP $\mathbb{S}_i$ [20]. Then, any admissible SSP heuristic can be used to obtain H_i, e.g., h^{lmc} [13] or h^{roc} [22]. On the other hand, $h^{\text{c-pom}}$ is embedded in i^2-dual-det. Furthermore, $h^{\text{c-pom}}$ does consider secondary cost constraints, and thanks to the coupling between itself and LP 1, the secondary cost constraints can be considered more effectively than by any cost projection approach, since the heuristic can take into account the secondary costs incurred by the policy being considered.

Infeasible Problems We are interested in feasible problems for this paper, but it is worth noting the behaviour of the algorithms when the problem is infeasible. LAny must iterate through every proper policy in stage 2 before terminating without an incumbent, and can then declare the problem unsolvable. i^2-dual-det, as long as it does not use alg. 1, can terminate and report the problem as unsolvable as soon as one of its partial CSSPs is unsolvable. In the worst case i^2-dual-det must expand every state and solve the partial CSSPs for each iteration, but with an informative heuristic the infeasible partial CSSP will occur much sooner. E.g., on a Constrained Exploding Blocks World problem (sec. 9) with 3 blocks that was made infeasible by its secondary cost constraints, i^2-dual-det determined that the problem is infeasible within a second, and LAny was stopped before termination after enumerating 3000 policies over 30 mins.

Challenging Constraints Constraints in CSSPs can make solving them more challenging in practice than SSPs despite both having the same worst case complexity [1]. However, this is not true for all constraints, e.g., constraints that are trivially satisfiable do not yield challenging CSSPs. This is a clear parallel to probabilistically interesting problems [15] where problems are probabilistically uninteresting if their probabilities can be ignored and one still obtains a good quality policy, and respectively interesting if ignoring probabilities leads to poor policies for the SSP. We define three levels of how interesting constraints are w.r.t. their CSSPs: (trivial) the optimal policy for the SSP relaxation is feasible and automatically optimal for the CSSP; (linearisable) there is a linearisation of costs s.t. the optimal policy is feasible for the CSSP but need not be optimal, i.e., $\exists\lambda$ s.t. the optimal policy for the λ-SSP is feasible for the CSSP; (interesting) when a problem is not linearisable. Trivial constraints are uninteresting since they can be ignored during search. Being able to linearise constraints suggests that they have minimal interaction, i.e., we do not encounter the case where decreasing one cost function forces another to increase in a way that requires a trade-off between the two.

LAny handles trivial problems very efficiently, since it starts by solving the λ-SSP with $\lambda = \vec{0}$, which yields an optimal feasible policy π for the CSSP, so LAny can terminate and return π. If a problem is linearisable and strong duality holds, then LAny may find the optimal policy as an incumbent and can then prove its optimality in stage 1. If strong duality does not hold then LAny has to enter stage 2, but does not have to spend a lot of time if its incumbent is optimal or close to optimal. On interesting problems, it is impossible for LAny to find a feasible solution in stage 1, which forces LAny to enter stage 2 and enumerate policies with a range of Lagrangian costs before finding the optimal, which can take a long time. In contrast, i^2-dual-det is not affected by the type of CSSP being solved.

We present one result about the categorisation of problems: CSSPs with one secondary cost constraint are linearisable. This is true because we just need to find $\lambda = [\lambda_1]$ with λ_1 large enough so that the secondary cost determines the cheapest policy in the λ-SSP, i.e., when λ_1 is large enough an infeasible policy is strongly penalised and a feasible policy strongly rewarded.

9 Empirical Evaluation

In this section we experimentally compare our new algorithm i^2-dual-det with LAny [14], the current state-of-the-art for finding optimal deterministic policies for CSSPs; and MIP 1 [8], as a baseline. All algorithms are implemented in C++ and the code is available at [17]. For solving LPs and MIPs we use CPLEX Version 22.1.1. To decide which planner settings to use, we ran preliminary experiments testing various setting combinations over a subset of the benchmarks with three instances per problem, with 45 problem instances in total. We now list planner specific settings:

i^2-dual-det In the preliminary experiments we tested i^2-dual-det with combinations of ways to deal with big-M (SOS1 constraints and alg. 1 with different settings) and different relaxations of the intermediate partial CSSPs (items 1 to 3 in sec. 6). With the relaxations from item 1 and item 3 it solves 39-45 instances, otherwise 23, i.e., the most important factor was the method for relaxing intermediate partial CSSPs. We selected the following two best performers for the full experiments: (1) i^2-dual-det-M uses alg. 1 with $M_0 = 1, \kappa = 10$, and only makes intermediate partial CSSPs easier to solve with the optimality gap approach (item 3), with a value of 1.0; and (2) i^2-dual-det-SOS1 uses SOS1 constraints, solves the LP relaxation of intermediate partial CSSPs (item 1), and then solves intermediate MIPs with an optimality gap requirement of 0.2 (item 3). Note that i^2-dual-det always uses the $h^{\text{c-pom}}$ heuristic, which is embedded [22], and additionally uses h^{max} as a dead-end detector on CSSPs that have dead ends before the finite penalty transformation.

LAny In the preliminary experiments, we tested LAny [14] with combinations of the following three parameters: (1) different heuristics h^{lmc} [13] and h^{roc} with h^{max} as a dead-end detector [22]; (2) the number of iterations in stage 1 is either limited to n iterations, where n is the number of secondary cost constraints, or it is unrestricted; (3) we tested the exact version of the algorithm, as described in this paper, as well as the two approximate versions with "candidate pruning" [14]. The settings in (2) and (3) are suggested by the authors of LAny, and used in their experiments. With h^{lmc}, LAny solved 31-39 instances, with h^{roc} it solved 23-31 problems. The number if iterations in stage 1 did not have an effect on coverage. The approximate variants solved fewer instances fully, and performed slightly worse in the anytime setting. So, in the full experiments, we only consider the exact version of LAny with h^{lmc} and an unrestricted number of iterations in stage 1. We also run it with h^{pom} [22], which we discuss and summarise later. The error tolerances are set to 0.0001. LAny relies on an upper bound on each λ_i, which can be selected manually, but we use the automatic method used by the algorithm's authors in their experiments.

MIP 1 We implemented this as LP 1 with SOS1 constraints.

We tested these algorithms on a suite of CSSPs. All our domains are encoded in Probabilistic PDDL. Some of the problems we consider violate the reachability assumption, which we address by applying the fixed-cost penalty formulation for dead ends [21], where for each state we add an artificial action that leads to a goal with probability 1 but incurs a large fixed-cost penalty. Since the separate cost functions of the CSSP usually represent different resources with different units, we require fixed-cost penalties to be specified for each cost function [20]. We consider the following domains:

Search and Rescue (SAR) [20] There are multiple survivors scattered on an $n \times n$ grid, and the agent must navigate a drone to bring exactly one of the survivors to a safe location as quickly as possible. The agent can move faster or slower, but that uses more or less fuel respectively. SAR problems have the single secondary cost constraint that fuel consumption should not exceed a given threshold. The threshold is selected automatically per problem as a function of the fuel usage of a policy in the SSP relaxation. We also consider problems where these thresholds are halved, denoted $h = T$ when halved and $h = F$ when original. SAR problems have a single secondary cost constraint, so they are linearisable.

Elevators (Elev) [20] The agent must route e elevators through an n-floor building so that all passengers arrive at their destinations. There are w actively waiting passengers and h hidden passengers who have not yet pressed the call button. Each problem has a bound on all passengers' waiting and travel times, i.e., there are $2(w + h)$ constraints secondary cost functions and constraints; these bounds are determined automatically based on n. Over 90% of the problem instances we consider with $e = 2$ have trivial constraints. For $e = 1$ there is a mix of trivial, linearisable, and interesting problems.

Constrained Exploding Blocks World (ExBW) [20] Exploding Blocks World [5] has N blocks that must be stacked on top of each other in a particular arrangement, where each block is rigged with an explosive, which with a certain probability explodes and destroys the table or block that it is placed on. Nothing can be placed on top of destroyed blocks and tables, and destroyed blocks can not be moved. In the constrained version [20], the table can be fixed with a large primary cost, but blocks can not. The only secondary cost constraint is that the number of destroyed blocks is bounded by threshold c, which is selected manually to ensure the problem is feasible. We consider interesting arrangements of blocks from IPPC'08 which we specify by their id. This domain has a single secondary cost constraint, so all its problems are linearisable.

Constrained PARC Printer (PARC) [22] PARC printer is an IPC domain where the agent must print s pages using a modular printer with various components. In this version, the printer has c unreliable components in which a page can get jammed with probability 0.1. Once jammed, the component can not be used for the remainder of execution, and the relevant page must be reprinted. Using certain components incurs a secondary cost whose expected cost must be bounded by u, and the probability of failure must be bounded by f. We fix $s = 4$ and vary c, u, f. In the problems we consider, they are trivial when $f = 1$ and $u = \infty$, otherwise linearisable or interesting.

Constrained Tireworld (CTW) This new domain is based on Triangle Tireworld [15], where the agent must drive a car over a triangular shaped network of roads and cities. Each time the agent drives between cities it has a 50% chance of getting a flat tyre. If the agent has a spare, it can fix the flat, otherwise it is unable to move. Spare tyres can only be acquired in certain cities. In the constrained version we make c copies of the action for getting a spare tyre, where copy i incurs 1 unit of secondary cost i. Respectively, there are c constraints, one for each secondary cost. We use intermediate tireworld problems [18], so problems are specified by n, the size of the triangle, and d, the agent's starting distance from the goal. CTW is linearisable but, for any linearisation λ such that the optimal policy for λ-SSP is CSSP-feasible, there are many other λ-SSP-optimal policies that are not CSSP-feasible. The effect is that solving the λ-SSP is unlikely to yield a CSSP-feasible policy, so these problems will have the same properties as interesting problems with high probability.

For a given problem, a problem instance on Elev and SAR refers to a random instantiation of the problem, and on ExBW, PARC, and CTW refers to the fixed problem paired with a random seed that is passed to the planner. We ran each planner on 30 instances of each problem on a cluster of Intel Xeon 3.2 GHz CPUs. For each instance, the planner was limited to 30 minutes CPU time, 4GB memory, and one CPU core. We now present and discuss the results.

		MIP 1		LAny		i^2-dual-det-M		i^2-dual-det-SOS1	
	problem	cov.	runtime	cov.	runtime	cov.	runtime	cov.	runtime
SAR (n, h)	(4, F)	0		26	142.3±53.9	15	753.2±229.0	25	605.8±235.5
	(4, T)	0		26	179.0±99.7	1	494.3	1	309.0
	(5, F)	0		30	229.2±71.1	5	264.9±276.4	0	
	(5, T)	0		27	249.8±73.8	2	564.4±633.4	0	
Elev (e,w,h)	(1, 2, 2)	30	9.9±3.5	10	13.6±3.2	30	247.6±84.7	30	68.8±42.8
	(2, 1, 1)	30	5.2±0.2	30	9.7±2.3	24	667.2±199.9	30	118.8±34.5
	(2, 1, 2)	30	275.9±103.9	29	125.4±17.9	0		0	
	(2, 2, 1)	30	196.2±118.3	28	64.5±12.8	1	1260.3	8	527.7±192.1
ExBW (id,N,c)	(4, 5, 0.17)	0		30	838.5±42.3	30	77.9±20.2	30	16.6±0.8
	(4, 5, 0.2)	0		30	0.2	30	10.6±1.1	30	1.6
	(5, 7, 0.1)	0		30	10.9±0.1	25	717.2±139.3	30	19.7±0.2
	(7, 8, 0.48)	0		30	1.8±0.1	27	559.6±144.5	30	35.3±2.6
PARC (c,f,u)	(1, 0.0, 1)	0		0		30	32.9±5.9	30	17.0±0.2
	(1, 0.0, ∞)	0		30	53.7±0.7	30	17.8±0.1	30	16.7±0.1
	(1, 0.1, 1)	0		0		30	69.9±24.2	25	121.6±123.1
	(1, 1.0, ∞)	0		30	20.7±0.4	0		0	
	(2, 0.0, 1)	0		0		30	32.8±6.2	30	16.7±0.1
	(2, 0.0, ∞)	0		30	151.6±1.7	30	17.3±0.1	30	16.1±0.2
	(2, 1.0, ∞)	0		30	44.7±0.5	0		0	
CTW (n,d,c)	(4, 4, 2)	30	371.9±58.0	30	349.0±1.4	30	110.6±16.3	30	29.6±2.1
	(4, 4, 4)	30	583.4±63.5	0		29	330.4±101.2	30	44.9±2.3
	(4, 4, 6)	18	949.4±91.6	0		29	513.9±79.7	30	57.0±2.1
	(4, 5, 2)	14	1621.7±60.6	0		30	385.9±26.7	30	141.8±14.7
	(4, 5, 4)	0		0		27	850.2±108.6	30	205.7±11.7
	(4, 5, 6)	0		0		25	1021.1±88.2	30	246.1±15.6
	(4, 6, 2)	0		0		0		30	1064.3±102.2
	(4, 6, 4)	0		0		0		25	1376.7±96.9
	(4, 6, 6)	0		0		0		28	1281.5±81.7

Table 1. Results for optimal planners. We present the coverage and the mean and 95% C.I. of runtime (secs) among the instances on which the planner converged in time. Maximum possible coverage is 30.

What is the best optimal planner? We present the coverage and runtimes of planner per problem in tab. 1. MIP 1 works well for small problems, but fails as the problems grow large, so we focus on comparing i^2-dual-det and LAny. LAny performs best on trivial problems, e.g., Elev with $e = 2$ and the trivial PARC problems. i^2-dual-det does best on interesting problems, e.g., interesting PARC and Elev with $e = 1$ problems. i^2-dual-det also does best on CTW problems. Recall that CTW is linearisable, but for any linearisation λ the λ-SSP has many optimal solutions that are not CSSP-feasible. This causes LAny to struggle because it is unlikely to find a CSSP-feasible policy in stage 1, and is forced to enter stage 2 and enumerate many policies before finding a CSSP-feasible one. On ExBW $id = 4, N = 5$, we see that i^2-dual-det is faster on $c = 0.17$, and LAny is faster on $c = 0.2$, i.e., i^2-dual-det performs better on problems with more challenging constraints. Incidentally, all our ExBW problems have strong duality except ExBW $id = 4, N = 5$, which intuitively has an effect on LAny, since weak duality requires the enumeration of more policies in stage 2. Strong and weak duality is not directly related to how interesting a problem is, but it does correlate with how well the λ^*-SSP approximates the CSSP. Comparing LAny using h^{lmc} and h^{pom} (omitted due to space) we have that: h^{lmc} is 1.5-3× faster on SAR and 1.5-38× on ExBW; h^{pom} has less coverage on PARC and CTW; and both have comparable results on Elev. LAny with h^{pom} still has better coverage than i^2-dual-det-SOS1 on SAR, is faster on the trivial Elev problems, but is slower than i^2-dual-det-SOS1 on all ExBW problems. Since h^{pom} is closer to $h^{\text{c-pom}}$, the heuristic used by i^2-dual-det, this suggests that LAny is outperforming i^2-dual-det on ExBW largely thanks to h^{lmc}. For SAR, this is only a small factor. Comparing i^2-dual-det-SOS1 and i^2-dual-det-M, the former dominates almost everywhere, except on large SAR problems and some PARC problems.

In summary, when considering optimal solutions, LAny is well suited to uninteresting CSSPs, i.e., trivial and linearisable CSSPs and generally underperforms on interesting problems and problems

			LAny		i^2-dual-det-M		i^2-dual-det-SOS1	
	problem	cutoff	cov.	opt. gap	cov.	opt. gap	cov.	opt. gap
SAR (n, h)	(4, F)	1800	30	0.015±0.016	30	0.046±0.065	30	0.003±0.003
	(4, T)	1800	30	0.010±0.016	30	0.720±0.157	22	0.153±0.144
	(5, F)	229	30	0.008±0.008	26	0.775±0.161	0	
	(5, T)	1800	30	0.002±0.003	30	0.933±0.091	0	
Elev (e,w,h)	(1, 2, 2)	9.9	2	0.000	1	1.000	17	0.017±0.014
	(2, 1, 1)	5.2	9	0.000	0		0	
	(2, 1, 2)	276	29	0.000	0		0	
	(2, 2, 1)	196	29	0.001±0.003	0		0	
ExBW (id,N,c)	(4, 5, 0.17)	16.6	30	0.005±0.001	0		30	0.002±0.001
	(4, 5, 0.2)	0.2	16	0.000	0		0	
	(5, 7, 0.1)	10.9	30	0.000	0		0	
	(7, 8, 0.48)	1.8	19	0.000	0		0	
PARC (c,f,u)	(1, 0.0, 1)	17.0	0		0		18	0.000
	(1, 0.0, ∞)	16.7	0		0		18	0.000
	(1, 0.1, 1)	69.9	0		21	0.000	21	0.000
	(1, 1.0, ∞)	20.7	14	0.000	0		0	
	(2, 0.0, 1)	16.7	0		0		18	0.000
	(2, 0.0, ∞)	16.1	0		0		10	0.000
	(2, 0.1, 1)	1800	30	0.055	0		27	0.026
	(2, 0.1, ∞)	1800	30	0.070	1	0.027	22	0.027
	(2, 1.0, 1)	1800	30	0.003±0.001	0		0	
	(2, 1.0, ∞)	44.7	16	0.000	0		0	
CTW (n,d,c)	(4, 4, 2)	29.6	0		9	1.000	17	0.000
	(4, 4, 4)	44.9	0		4	0.990±0.020	14	0.000
	(4, 4, 6)	57.0	0		17	1.000	13	0.000
	(4, 5, 2)	142	0		12	0.997±0.007	19	0.000
	(4, 5, 4)	206	0		14	1.000	13	0.000
	(4, 5, 6)	246	0		13	1.000	14	0.000
	(4, 6, 2)	1064	0		18	1.000	15	0.000
	(4, 6, 4)	1800	0		14	0.993±0.009	25	0.000
	(4, 6, 6)	1800	0		15	1.000	28	0.000

Table 2. Results for anytime planners. We show the cutoff (secs) per problem, the anytime coverage (number of instances for which the planner produces an incumbent before the cutoff), and the mean and 95% C.I. of runtime (secs) for covered instances. Maximum possible coverage is 30.

that are linearisable but with high probability the optimal solution for any λ-SSP is infeasible, such as CTW. In contrast, i^2-dual-det is not negatively affected by the complexity of the constraints and outperforms other planners in interesting CSSPs. Nonetheless, i^2-dual-det is unable to take advantage of simpler constraints structures and it is outperformed by LAny in the uninteresting CSSPs. MIP 1 has a similar tradeoff as i^2-dual-det; however, it is unable to scale up to large problems since, unlike i^2-dual-det and LAny, it does not perform heuristic search in the planning search space and it is forced to consider all states and actions.

Are anytime planners more practical than optimal planners? It would be unfair to compare the anytime algorithms only within the scope of solving problems optimally, so we look at how good their incumbent solutions are before the optimal planners fully converge in tab. 2. For each problem p we consider the optimal planners will full coverage in tab. 1, and among these we define m_p to be the minimum mean time to solve. If no planner attains full coverage we set m_p to the deadline time. In tab. 2, the anytime coverage is the number of instances where LAny has an incumbent solution before m_p. We define the optimality gap of a solution as $|ub - lb|/|ub|$.

LAny often finds incumbent solutions before the cutoff with very tight optimality gaps. The important exceptions are CTW, which is linearisable but an optimal policy for its λ-SSPs is very likely CSSP-infeasible; and non-trivial PARC problems where LAny does not find any incumbent solutions before the cutoff. On Elev $e = 1, w = 2, h = 2$ LAny finds incumbent solutions in time for trivial instances, and for the non-trivial instances it finds none. This suggests that on the problems where LAny performs poorly as an optimal planner, its anytime capabilities can not redeem it. i^2-dual-det-SOS1 has higher coverage and better optimality gaps than i^2-dual-det-M except in SAR and CTW, where i^2-dual-det-M has higher coverage but with a large optimality gap; this can be primarily attributed to i^2-dual-det-M's optimality gap requirement of 1, compared to i^2-dual-det-SOS1's 0.2.

10 Conclusion and Future Work

Dolgov and Durfee [8]'s MIP for finding optimal deterministic policies on CMDPs can be generalised to solve CSSPs, but we showed that there is an important technical detail that must be considered: the automatic derivation of the constant M relies on the inexistance of improper policies in CMDPs, which is not true for the more general model of CSSPs. This is an important issue that can not be swept under the rug since poorly chosen M can affect correctness. We addressed this issue with new theory and algorithms.

We also introduced i^2-dual-det, a new algorithm for finding optimal deterministic policies, obtained by adapting the state-of-the-art algorithm for optimal stochastic policies i^2-dual [22] with the approaches we developed earlier. We enhanced the base algorithm with various MIP solving techniques, which in the spirit of iLAO*, solve the intermediate partial CSSPs approximately. Not only do these enhancements improve performance, but one particular enhancement endows i^2-dual-det with anytime capabilities.

We compared our algorithm i^2-dual-det with LAny [14], the state-of-the-art for finding deterministic policies for CSSPs. Theoretical differences include the kinds of heuristics that can be used, and that LAny has to do a lot more work than i^2-dual-det to prove infeasibility. We compared performance empirically on a set of benchmarks and found that the algorithms perform well on different classes of problems: LAny is the fastest algorithm for trivial and linearisable problems, whereas i^2-dual-det is the fastest for interesting problems and linearisable problems where, for each λ-SSP whose optimal policy is CSSP-feasible, there are also many λ-SSP-optimal policies that are not CSSP-feasible.

In parallel with this work, Steinmetz et al. [19] adapted i^2-dual to find deterministic policies for SSPs constrained by policy properties, which generalise CSSPs. Their approach is equivalent to our i^2-dual-det with big-M constraints, where M is manually selected. The techniques we provide in sec. 5 for automatically dealing with M can be applied to their algorithm, and similarly, the techniques in sec. 6 for speeding up i^2-dual-det can also be applied to their algorithm.

Our future agenda includes scaling heuristic search methods to larger problems by developing domain-independent heuristics specifically for deterministic policies. We considered i^2-dual-det with the $h^{\text{c-pom}}$ heuristic, which is admissible for stochastic policies; but, the cost of stochastic policies is a loose lower bound on deterministic ones, so there is room to make the heuristics more informative. Another direction to improve scalability is to develop admissible heuristics for λ-SSPs that do not rely on cost-projection thus taking into account the interaction between cost functions.

Another item in our future agenda is to combine our techniques for finding deterministic policies with techniques for multi-objective planning [10, 6] to compute the Pareto coverage set for SSPs and CSSPs, i.e., the finite set of non-dominated deterministic policies.

11 Acknowledgements

This research/project was undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI), which is supported by the Australian Government.

References

[1] E. Altman. *Constrained Markov Decision Processes: Stochastic Modeling*. Routledge, 2021. ISBN 9781315140223. doi: https://doi.org/10.1201/9781315140223.

[2] E. Beale and J. Tomlin. Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. *Operational Research*, 69:447–454, 01 1969.

[3] D. P. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019. ISBN 9781886529397.

[4] D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 1991.

[5] D. Buffet. International planning competition uncertainty part: Benchmarks and results, 2008.

[6] D. Chen, F. Trevizan, and S. Thiébaux. Heuristic Search for Multi-Objective Probabilistic Planning. In *Proc. of 37th AAAI Conference on Artificial Intelligence*, 2023.

[7] F. d'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108, 1963. URL https://pubsonline.informs.org/doi/abs/10.1287/mnsc.10.1.98.

[8] D. A. Dolgov and E. H. Durfee. Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[9] E. A. Feinberg. Constrained discounted markov decision processes and hamiltonian cycles. *Mathematics of Operations Research*, 2000.

[10] F. Geisser, P. Haslum, S. Thiébaux, and F. Trevizan. Admissible Heuristics for Multi-Objective Planning. In *Proc. of 32nd Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2022.

[11] F. Geißer, G. Povéda, F. Trevizan, M. Bondouy, F. Teichteil-Königsbuch, and S. Thiébaux. Optimal and heuristic approaches for constrained flight planning under weather uncertainty. *Proceedings of the International Conference on Automated Planning and Scheduling*, 2020.

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

[13] M. Helmert and C. Domshlak. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? *Proc. of 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2009.

[14] S. Hong and B. C. Williams. An anytime algorithm for constrained stochastic shortest path problems with deterministic policies. *Artificial Intelligence*, 2023.

[15] I. Little, S. Thiebaux, et al. Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*, 2007.

[16] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 2013.

[17] J. Schmalz and F. Trevizan. Code and Benchmarks for ECAI 2024 paper "Finding Optimal Deterministic Policies for Constrained Stochastic Shortest Path Problems", July 2024. URL https://doi.org/10.5281/zenodo.12714199.

[18] J. Schmalz and F. Trevizan. Efficient constraint generation for stochastic shortest path problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[19] M. Steinmetz, S. Thiébaux, D. Höller, and F. Teichteil-Königsbuch. Explaining the Space of SSP Policies via Policy-Property Dependencies: Complexity, Algorithms, and Relation to Multi-Objective Planning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 2024.

[20] F. Trevizan, S. Thiébaux, P. Santana, and B. Williams. Heuristic Search in Dual Space for Constrained Stochastic Shortest Path Problems. In *Proc. of 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2016.

[21] F. Trevizan, F. Teichteil-Königsbuch, and S. Thiébaux. Efficient Solutions for Stochastic Shortest Path Problems with Dead Ends. In *Proc. of 33rd Int. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[22] F. Trevizan, S. Thiébaux, and P. Haslum. Occupation Measure Heuristics for Probabilistic Planning. In *Proc. of 27th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2017.