I-dual: Solving Constrained SSPs via Heuristic Search in the Dual Space

Felipe Trevizan **Sylvie Thiébaux**
Data61, CSIRO
Australian National University
first.last@anu.edu.au

Pedro Santana **Brian Williams**
MERS Group
Massachusetts Institute of Technology
{psantana,williams}@mit.edu

Abstract

We consider the problem of generating optimal stochastic policies for Constrained Stochastic Shortest Path problems, which are a natural model for planning under uncertainty for resource-bounded agents with multiple competing objectives. While unconstrained SSPs enjoy a multitude of efficient heuristic search solution methods with the ability to focus on promising areas reachable from the initial state, the state of the art for constrained SSPs revolves around linear and dynamic programming algorithms which explore the entire state space. In this paper, we present i-dual, the first heuristic search algorithm for constrained SSPs. To concisely represent constraints and efficiently decide their violation, i-dual operates in the space of dual variables describing the policy occupation measures. It does so while retaining the ability to use standard value function heuristics computed by well-known methods. Our experiments show that these features enable i-dual to achieve up to two orders of magnitude improvement in run-time and memory over linear programming algorithms.

1 Introduction

Stochastic Shortest Paths problems (SSPs) are widely used models for planning under uncertainty. Given an initial state, a set of goal states, actions with probabilistic outcomes, and an action cost function, the objective is to find a policy that minimises the expected cost of reaching the goal from the initial state. For SSPs, the optimal policies are deterministic mappings from states to actions and can be efficiently computed using heuristic search algorithms [Mausam and Kolobov, 2012]. These algorithms focus the search on promising regions of the state space by using the guidance of an admissible heuristic. This focused search sets these algorithms apart from linear programming formulations and dynamic programming algorithms, such as Value and Policy Iteration, which explore the entire state space.

In many application domains, SSPs have limited value, as multiple competing performance criteria need to be considered and are difficult to encapsulate in a single scalar cost function [Undurti and How, 2010]. For example, in a UAV search and rescue mission, as many targets as possible need to be found as fast as possible, whilst minimising battery usage and the probability of reaching dangerous areas. A natural approach to deal with such situations is to optimise a primary cost function, and constrain the others [Altman, 1999]. This leads to *constrained SSPs* which augment SSPs with upper bounds on the expected value of the secondary costs.

In this paper, we present i-dual, the first heuristic algorithm for constrained SSPs. Similarly to unconstrained planning algorithms, i-dual incrementally builds the search space reachable from the initial state by repeatedly expanding the fringe states reachable under the current best policy, and updating that policy using linear programming. By using the dual LP formulation [D'Epenoux, 1963], i-dual can easily represent the secondary cost constraints and ensures that the resulting policy satisfies them. Our empirical evaluation shows that i-dual successfully leverages the heuristics over the primary and secondary cost functions to prune suboptimal and infeasible solutions, and is able to solve problems up to two orders of magnitude larger than the regular dual LP approach. Moreover, we show how non-admissible heuristics can be used by i-dual to trade-off optimality with scalability and quickly compute policies that still respect the constraints, an option not offered by the dual LP approach.

The paper is organised as follows: Section 2 presents the background on SSPs and constrained SSPs; i-dual, our novel algorithm, is introduced in Section 3; the empirical evaluation of i-dual is presented in Section 4; and we conclude in Section 5. Further details can be found in [Trevizan *et al.*, 2016].

2 Background

A **Stochastic Shortest Path problem** (SSP) [Bertsekas and Tsitsiklis, 1991] is a tuple $\langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C \rangle$ in which: S is the set of states; s_0 is the initial state; $\mathsf{G} \subseteq \mathsf{S}$ is the set of goal states; A is the set of actions; $P(s'|s,a)$ is the probability that s' is reached after applying action a in state s; and $C(s,a) \in \mathbb{R}^*_+$ is the cost of applying action a in state s. We represent by $\mathsf{A}(s)$ the actions applicable in state s and we assume that a goal state is reachable from every state s.

A solution to an SSP is a policy π, i.e., a mapping from states to actions. If a policy π can be executed from s_0 to a goal state for any probabilistic outcome of the actions (i.e., no replanning is needed) then π is a closed policy. A policy can also be deterministic or stochastic. The former maps a state

s to one action $a \in \mathsf{A}(s)$ and the latter maps a state s to a probability distribution over $\mathsf{A}(s)$. We denote by $\pi(s,a)$ the probability of executing a in s.

The optimal solution for an SSP is a closed policy π^* that minimises the total expected cost of reaching the goal from s_0. For SSPs, π^* might not be unique and there always exists at least one optimal policy that is deterministic. The optimal solution of an SSP can be uniquely described by the optimal value function V^*, i.e., a function from S to $\mathbb{R}_+$ representing the minimum total expected cost to reach the goal from each state s [Bertsekas and Tsitsiklis, 1996]. V^* can be computed iteratively, using linear programming, and through heuristic search [Mausam and Kolobov, 2012]. We refer to these algorithms that explicitly compute V^* as primal-space algorithms. An admissible heuristic H for primal-space algorithms is any lower bound on V^*.

An alternative to primal-space algorithms is to solve the linear program over the dual space [D'Epenoux, 1963] presented in (LP 1) in which: the optimisation variables $x_{s,a}$, known as occupation measure, represent the expected number of times action $a \in \mathsf{A}(s)$ is executed in s; $in(s) = \sum_{s' \in \mathsf{S}, a \in \mathsf{A}(s')} x_{s',a} P(s|s',a)$; and $out(s) = \sum_{a \in \mathsf{A}(s)} x_{s,a}$.

$$\min_{x} \sum_{s \in \mathsf{S}, a \in \mathsf{A}(s)} x_{s,a} C(s,a) \quad \text{s.t.} \quad \text{(C1)} - \text{(C4)} \qquad \text{(LP 1)}$$

$$x_{s,a} \geq 0 \qquad \forall s \in \mathsf{S}, a \in \mathsf{A}(s) \qquad \text{(C1)}$$
$$out(s) - in(s) = 0 \qquad \forall s \in \mathsf{S} \setminus (\mathsf{G} \cup \{s_0\}) \qquad \text{(C2)}$$
$$out(s_0) - in(s_0) = 1 \qquad \text{(C3)}$$
$$\textstyle\sum_{s_g \in \mathsf{G}} in(s_g) = 1 \qquad \text{(C4)}$$

This dual formulation can be interpreted as a *flow problem*, where: $in(s)$ and $out(s)$ define the expected flow entering and leaving the state s, respectively; (C2) is the flow conservation principle; and (C3) and (C4) define, respectively, the source (s_0) and the sinks (goal states). The objective function of (LP 1) captures the minimisation of the total expected cost to reach the goal from s_0. Once we have the solution x^* of (LP 1), the optimal policy is $\pi^*(s,a) = x^*_{s,a}/out(s)$ and is guaranteed to be deterministic, i.e., $x^*_{s,a} > 0$ for exactly one action $a \in \mathsf{A}(s)$ for all s such that $out(s) > 0$.

A **constrained SSP** (C-SSP) is an SSP with $k+1$ cost functions in which one cost function is optimised while the remaining k costs are constrained by an upper bound. Formally, a C-SSP is the tuple $\langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, \vec{C}, \vec{\hat{c}} \rangle$ where $\mathsf{S}, s_0, \mathsf{G}, \mathsf{A}$, and P are defined as in the SSPs and $\vec{C} = [C_0, \ldots, C_k]$ is the cost functions vector and $\vec{\hat{c}} = [\hat{c}_1, \ldots, \hat{c}_k]$ is the cost upper bound vector ($\hat{c}_j > 0, \ \forall j$). We refer to C_0 as *primary* cost and all other cost functions as *secondary* costs.

The constraints on costs C_j, for $j \in \{1, \ldots, k\}$, are on expectation, i.e., given a policy π, the *expected* value of C_j over all executions of π from s_0 to the goal must be less or equal to $\hat{c}_j$. Even though an upper bound on the maximum incurred cost (as opposed to its expectation) might be desired, such bounds make most SSPs infeasible because they require that none of the possible executions of π violate the constraints.

The optimal solution to a C-SSP is defined as the solution of (LP 1) subject to the additional constraint C5 and replacing C by the primary cost function C_0 in the objective function.

$$\sum_{s \in \mathsf{S}, a \in \mathsf{A}(s)} x_{s,a} C_j(s,a) \leq \hat{c}_j \qquad \forall j \in \{1, \ldots, k\} \qquad \text{(C5)}$$

In contrast to SSPs, there is no guarantee that the optimal policy for C-SSPs is deterministic. Nevertheless, the complexity of finding the (potentially stochastic) optimal policy for C-SSPs is polynomial in the number of reachable states as in the case of SSPs [Altman, 1999].

The occupation measure space is interesting for C-SSPs because, the total expected j-th cost of following x from s_0 to a goal can be easily computed by $\sum_{s,a} x_{s,a} C_j(s,a)$. In contrast, primal-space algorithms have to represent and compute the fixed-point solution for each value function V_j associated with C_j ($j \in \{0, \ldots, k\}$). Moreover, the set of costs computed need to be those for a single policy, and imposing this constraint in the primal-space is non-trivial.

Although the dual approach is tractable, a significant computational cost required is that all the reachable states from s_0 must be encoded in the LP, even if only a fraction of them are needed to compute π^*. Furthermore, deriving a non-zero lower bound for the occupation measure x is not trivial because $x_{s,a} > 0$ if and only if $\pi^*(s,a) > 0$ for at least one optimal policy π^*. In fact, the authors are not aware of any domain-independent bounds for $x_{s,a}$.

3 i-dual

To overcome the lack of lower bounds for $x_{s,a}$, we use a heuristic vector over costs $\vec{H} = [H_0, \ldots, H_k]$, where $H_j \colon \mathsf{S} \to \mathbb{R}_+$ for all $j \in \{0, \ldots, k\}$ is a heuristic for the expected value of C_j (i.e, a primal-space heuristic). A heuristic H_j is admissible for C_j if it is a lower bound for the j-th total expected cost and it can be easily derived by relaxing the C-SSP into an unconstrained SSP as a range of heuristics exist for SSPs [Mausam and Kolobov, 2012]. Formally, given a C-SSP $\langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, \vec{C}, \vec{\hat{c}} \rangle$ and $j \in \{0, \ldots, k\}$, let $\mathbb{S}_j$ be the SSP $\langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C_j \rangle$, then the optimal value function V^* for $\mathbb{S}_j$ is an admissible heuristic for C_j. Therefore, any admissible heuristic H for $\mathbb{S}_j$ is also admissible for C_j.

Our novel approach consists in defining incrementally larger *partial problems* starting from s_0, using artificial goals to represent non-explored areas of the occupation measure space. Artificial goals are prioritised using their heuristic values, incurred as a one-time *terminal cost* when the artificial goal state is first reached. Each partial problem is thus represented as a C-SSP with terminal costs $\langle \hat{\mathsf{S}}, s_0, \hat{\mathsf{G}}, \hat{\mathsf{A}}, P, \vec{C}, \vec{\hat{c}}, \vec{H} \rangle$, where: $\langle \hat{\mathsf{S}}, s_0, \hat{\mathsf{G}}, \hat{\mathsf{A}}, P, \vec{C}, \vec{\hat{c}} \rangle$ is a regular C-SSP, $\hat{\mathsf{S}} \subseteq \mathsf{S}$, $\mathsf{G} \cap \hat{\mathsf{S}} \subseteq \hat{\mathsf{G}}$, $\hat{\mathsf{A}} \subseteq \mathsf{A}$, and $\vec{H}$ is the heuristic vector. The dual linear program for C-SSPs with such terminal costs is presented in LP 2.

$$\min_{x} \sum_{s \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s)} x_{s,a} C_0(s,a) + \sum_{s_g \in \hat{\mathsf{G}}} in(s_g) H_0(s_g)$$
$$\text{s.t.} \quad \text{(C1)} - \text{(C4)}, \text{(C6)} \qquad \text{(LP 2)}$$

$$\sum_{s \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s)} x_{s,a} C_j(s,a) + \sum_{s_g \in \hat{\mathsf{G}}} in(s_g) H_j(s_g) \leq \hat{c}_j \quad \forall j \qquad \text{(C6)}$$

```
1  I-DUAL(C-SSP ⟨S, s_0, G, A, P, C⃗, c⃗̂⟩, and vector H⃗)
2  begin
3      Ŝ ← {s_0}; F ← {s_0}; F_R ← {s_0}
4      while F_R ≠ ∅ do
5          N ← {s' ∉ Ŝ | ∃s ∈ F_R, a ∈ A(s), P(s'|s,a) > 0}
6          Ŝ ← Ŝ ∪ N
7          F ← (F \ F_R) ∪ (N \ G)
8          Ĝ ← F ∪ (G ∩ Ŝ)
9          Â ← {a | ∃s ∈ Ŝ \ F, a ∈ A(s)}
10         x ← SOLVE(LP 2 for ⟨Ŝ, s_0, Ĝ, Â, P, C⃗, c⃗̂, H⃗⟩)
11         F_R ← {s ∈ F | in(s) > 0}
12     for (s,a) s.t. x_{s,a} > 0 do π(s,a) ← x_{s,a}/out(s)
13     return π
```

Algorithm 1: i-dual algorithm for solving C-SSPs using incrementally larger LPs over the occupation measure space and heuristic search over the value function space.

I-dual is depicted in Algorithm 1 and the key elements that vary from one iteration of the algorithm to the next are: the set $\hat{\mathsf{S}}$ of states considered so far; the set $\mathsf{F} \subseteq \hat{\mathsf{S}}$ of fringe states (i.e., the unexpanded states in $\hat{\mathsf{S}}$); the fringe states $\mathsf{F}_R \subseteq \mathsf{F}$ reachable from s_0 by following the policy encoded by x; and the set $\hat{\mathsf{G}}$ of artificial goals for the current partial problem. At each iteration, i-dual computes the set N of all the new states reachable from F_R by applying one action (line 5), and updates the set of states, the fringe, and the artificial goals accordingly (lines 6-8). When F_R is empty, a closed policy has been found, that is, all of the flow injected into the system on s_0 is absorbed by goals of the original problem.

Since i-dual always enforces the cost constraints (C6), the returned policy π (if any) is always feasible regardless of the heuristic vector used and its admissibility. Moreover, if all the heuristics in $\vec{H}$ are admissible, then π is optimal:

Theorem 1 ([Trevizan *et al.*, 2016])**.** *Given a C-SSP* $\mathbb{C} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, \vec{C}, \vec{c} \rangle$ *and a vector of admissible heuristics* $\vec{H}$*, the policy* π *returned by i-dual is an optimal policy for* $\mathbb{C}$*.*

Similarly to primal-space heuristic search algorithms for SSPs, i-dual is complete but suboptimal when the heuristics $H_1, \ldots, H_k$ for the secondary costs are admissible but the heuristic H_0 for the primary cost is not. When any secondary heuristic is not admissible, i-dual is incomplete: it might not find a solution even if one exists because the non-admissible heuristic can incorrectly indicate that a feasible solution violates the constraints.

I-dual can be viewed through two different prisms: as a heuristic search algorithm and as a column and row generation algorithm. In the former, an Artificial Intelligence point of view, i-dual is analogous to A* where the cost $g(n)$ to reach an artificial goal (fringe) n from s_0 is computed using LP 2. The LP handles the search in the cyclic space of occupation measures to find a solution that minimises the expected value of $g(n) + H_0(s)$. As in A*, the selected fringe states (F_R in Algorithm 1) are expanded and the search continues until a solution is found.

Alternatively, from an Operations Research point of view, i-dual is a column and row generation algorithm. That is, at each iteration, new variables $x_{s,a}$ are added (columns) as well as new flow conservation constraints (rows) for the newly expanded states, and their corresponding occupation measures. The heuristics $\vec{H}$ prioritise the candidate columns and the solution of LP 2 indicates what columns should be added, namely, $x_{s,a}$ for all $s \in \mathsf{F}_R$ and $a \in \mathsf{A}(s)$. Columns are added in batch because, if a is applied in s, then all states s' s.t. $P(s'|s,a) > 0$ need to be considered in order to obtain a closed policy.

The column and row generation point of view is not only theoretically relevant, it also increases the performance of i-dual. The overhead of building the LPs in line 12 (Algorithm 1) is minimised by keeping only one LP in memory and adding columns and rows to it. More importantly, this approach can take advantage of "warm starts", i.e., to solve the new LP by reusing the solution of the previous one.

4 Empirical Evaluation

We empirically evaluated i-dual and the dual LP (LP 1 with the additional constraint C5) in three different domains:

Search and rescue: an $n \times n$ grid navigation problem where the goal is to find one survivor, board her on the vehicle and bring her to a safe location as fast as possible. The constraint is to keep the expected fuel consumption under a certain threshold. The location of one survivor is known a priori; however, some other locations (selected at random) have 0.05, 0.1, or 0.2 probability of also having a survivor. Thus, the planner has to trade off fuel, exploration of unknown survivors, and time to rescue. The parameters for the SAR problem generator is n, the distance to the known survivor (d), and the density of potential survivors (r).

Elevators: a 20-storey building with e elevators in which w persons are known to be waiting for an elevator and h persons are expected to arrive and request an elevator. Each "hidden" person arrives at the elevator in the next time step with probability 0.75. The origin and destination of all persons are known a priori. In this domain, the planner has to minimize the number of elevators movements (actions) while satisfying the maximum waiting and travelling time of each person; and

Exploding blocks world: an extension of the domain with same name from IPPC [Bryce and Buffet, 2008] in which the expected number of blocks explosion is constrained through an upper bound. This extension does not have unavoidable dead ends due to the introduction of the action *fix-table* that restores the table to its original condition in case a block destroyed it. For these problems, the primary objective is to minimise the action costs. The problem instances are those of the IPPC augmented with the constraint and the fix-table action.

The heuristic vector $\vec{H}$ used by i-dual is denoted as (X, Y) representing $[X, Y, Y, \ldots, Y]$ where the heuristics for C_j is computed over the all-outcomes determinisation of the SSP with C_j as cost function (Section 3). We consider the following domain-independent heuristics: always zero (h_0), maximal ($h_{\max}$), additive (h_{add}), and lm-cut (h_{lmc}). The complete empirical evaluation, description of the domains and methodology can be found in [Trevizan *et al.*, 2016].

Table 1 shows the average cpu-time, number of states explored and percentage of the cpu-time spent encoding LP 2,

	Problem	Planner	Avg cpu time (s)	Avg # of vis. states	Average % time computing Enc.	$\vec{H}$	x
SAR (n, r, d)	$4, 0.5, 4$	dual-lp	622.6	102,928	7.2%	–	92.4%
		h_{lmc},$h_{\max}$	220.2	4,129	1.6%	1.3%	96.3%
		h_{add},h_{add}	142.4	3,693	1.9%	0.4%	96.7%
	$5, 0.25, 4$	dual-lp	128.4	53,075	21.3%	–	77.9%
		h_{lmc},$h_{\max}$	30.6	1,459	3.1%	7.4%	88.2%
		h_{add},h_{add}	15.6	1,263	3.7%	2.5%	92.0%
	$5, 0.75, 3$	h_{lmc},$h_{\max}$	444.2	6,015	1.4%	1.8%	95.8%
		h_{add},h_{add}	200.8	4,604	1.9%	0.7%	96.3%
Elevators (e, h, w)	$1, 2, 2$	dual-lp	1.7	5,317	43.1%	–	52.5%
		$h_{\max}$,$h_{\max}$	11.2	1,478	5.4%	47.4%	43.9%
		h_{lmc},$h_{\max}$	91.6	1,255	0.4%	95.3%	3.7%
		h_{lmc},h_0	106.5	1,258	0.4%	95.9%	3.1%
		h_{add},h_0	2.6	529	6.5%	32.5%	55.2%
		h_{add},h_{add}	3.0	486	5.1%	60.2%	30.1%
	$2, 1, 2$	dual-lp	375.0	117,819	6.2%	–	93.5%
		h_{add},h_0	18.7	1,034	5.4%	4.1%	88.2%
		h_{add},h_{add}	8.6	957	8.3%	15.4%	72.4%
Exploding BW	p01	dual-lp	849.2	342,650	2.5%	–	97.2%
		h_{lmc},$h_{\max}$	9.7	3,466	10.5%	4.0%	83.8%
		h_{add},h_{add}	9.0	3,388	10.8%	1.0%	86.4%
	p02	dual-lp	1603.1	359,343	1.4%	–	98.4%
		h_{lmc},$h_{\max}$	865.1	19,730	1.7%	0.2%	97.5%
		h_{add},h_{add}	665.3	17,302	1.9%	0.0%	97.5%
	p04	h_{lmc},$h_{\max}$	1446.1	30,689	1.7%	0.3%	97.4%
		h_{add},h_{add}	1153.1	28,576	1.8%	0.1%	97.6%

Table 1: Cpu-time, number of states explored, and perc. of the cpu-time spent encoding LP 2, computing heuristics $\vec{H}$, and solving LP 2 for increasingly harder problems. Only planners that obtained 100% coverage are shown. h_X, h_Y represents i-dual(h_X, h_Y). The dimension of $\vec{H}$ for the search and rescue (SAR) and exploding blocks world is 2 for all problems and, for elevators problems, it is 9 and 7, respectively.

computing the vector of heuristics $\vec{H}$, and solving LP 2 for selected problems of each domain. For SAR and elevators, each problem number (second column) represents a different parametrization of the generator and the planners were evaluated by solving 30 randomly generated problems for each parametrization. For the augmented exploding blocks world, each entry represents the problem number in IPPC'08 and planners were evaluated by solving each problem 30 times using different random seeds. Only planners that obtained 100% coverage (i.e., found a closed policy from s_0 for all the 30 runs in each entry) are shown in Table 1.

For SAR, the dual LP fails to scale up and is unable to solve even trivial problems (e.g., for $n = 4$, $d = 1$ and $r = 0.75$) due to the large reachable state space that needs to be explicitly constructed even if only a small fraction of it is needed to compute the optimal policy. As shown in Table 1, i-dual is able to explore at least one order of magnitude less states than the dual LP, allowing it to scale up to larger problems. For the elevator problems, the dual LP dominates all i-dual parametrisations when the reachable state space is small because, in this case, it is feasible to encode the complete dual LP. The admissible parametrisations of i-dual were also dominated by the dual LP in the medium size instances because of the large number of cost constraints ($1 + 2(w + h)$ constraints) and thus the dimension of $\vec{H}$ (see column 7 of Table 1). $h_{\max}$ and h_{lmc} also provided weak guidance for this domain not paying off the overhead to compute them. In contrast, the non-admissible h_{add} was able to offer good guidance and allowed i-dual to solve problems overcoming the overhead of the large number of heuristic calls per state. For the exploding blocks world, the dual LP is unable to solve problems with more than 5 blocks while i-dual is able to solve problems with up to 8 blocks. Furthermore, i-dual is two orders of magnitude faster than the dual LP for the problems that the latter can solve. As shown in Table 1, i-dual obtains this performance by using the heuristics to prune a large portion of the state space.

5 Conclusion, Related and Future Work

I-dual combines heuristic search and the dual LP formulation of SSPs to optimally solve constrained SSPs resulting, to the best of our knowledge, in the first heuristic search algorithm for constrained SSPs (C-SSPs). I-dual is able to efficiently encode and enforce secondary cost constraints, and to focus the search on regions that are relevant to the optimal stochastic policy, using (admissible) value function heuristics. This results in up to two orders of magnitude improvements in run-time and memory when compared to C-SSP algorithms based on linear programming alone. Furthermore, i-dual is able to trade-off optimality with scalability and quickly compute policies that still respect the constraints, an option not offered by the primal linear programming approach.

In terms of related work, Altman (1999) was one of the first to research linear and dynamic programming algorithms for a range of constrained Markov Decision Processes. Dolgov and Durfee (2005) extend the work of Altman to the generation of deterministic policies for constrained MDPs. None of these algorithms are capable of using a heuristic to guide their search. Santana, Thiébaux, Williams (2016) propose RAO*, an extension of AO* for finite-horizon POMDPs with *chance constraints* (i.e., bounds on the probability of a constraint being violated) that uses heuristics to guide its search in the primal space. Note that, chance constraints can be encoded as constraints over expectation if we assume that their their violation cause policy termination. Trevizan and Veloso (2012) introduce an algorithm for *unconstrained* SSPs that shares with i-dual the usage of subproblems with artificial goals but performs search in the primal space. Our future work agenda includes combining both algorithms to obtain a version of i-dual for planning and execution that could offer guarantees regarding constraint violation during execution.

Lastly, Teichteil, Sprauel, and Kolobov (2012, 2014) consider the generation of optimal stochastic policies for discounted MDPs with path constraints expressed in probabilistic computational tree logic (pCTL). They describe iterative linear and dynamic programming algorithms that improve on algorithms used in the probabilistic model-checking community [Kwiatkowska and Parker, 2013, Baier *et al.*, 2004]. Having an efficient algorithm for C-SSPs which could be used or extended to deal with path constraints in probabilistic temporal logic is one of the motivation that lead to i-dual, and this extension is an item on our future work agenda.

Acknowledgements

This research was funded by AFOSR grant FA2386-15-1-4015. We thank Patrik Haslum for fruitful discussions and the anonymous reviewers for their helpful comments.

References

[Altman, 1999] Eitan Altman. *Constrained Markov Decision Processes*, volume 7. CRC Press, 1999.

[Baier *et al.*, 2004] Christel Baier, Marcus Größer, Martin Leucker, Benedikt Bollig, and Frank Ciesinski. Controller synthesis for probabilistic systems. In *Proc. Int. Conf. on Theoretical Computer Science (TCS)*, 2004.

[Bertsekas and Tsitsiklis, 1991] D.P. Bertsekas and J.N. Tsitsiklis. An Analysis of Stochastic Shortest Path Problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

[Bertsekas and Tsitsiklis, 1996] D. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Bryce and Buffet, 2008] D. Bryce and O. Buffet. 6th International Planning Competition: Uncertainty Track. In *3rd Int. Probabilistic Planning Competition (IPPC-ICAPS'08)*, 2008.

[D'Epenoux, 1963] F. D'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10:98–108, 1963.

[Dolgov and Durfee, 2005] Dmitri A. Dolgov and Edmund H. Durfee. Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005.

[Kwiatkowska and Parker, 2013] Marta Z. Kwiatkowska and David Parker. Automated verification and strategy synthesis for probabilistic systems. In *Int. Symp. on Automated Technology for Verification and Analysis*, 2013.

[Mausam and Kolobov, 2012] Mausam and Andrey Kolobov. *Planning with Markov Decision Processes*. Morgan&Claypool, 2012.

[Santana *et al.*, 2016] Pedro Santana, Sylvie Thiébaux, and Brian Williams. RAO*: an algorithm for chance constrained POMDPs. In *Proc. AAAI Conference on Artificial Intelligence*, 2016.

[Sprauel *et al.*, 2014] Jonathan Sprauel, Andrey Kolobov, and Florent Teichteil-Königsbuch. Saturated path-constrained MDP: planning under uncertainty and deterministic model-checking constraints. In *Proc. AAAI Conf. on Artificial Intelligence*, 2014.

[Teichteil-Königsbuch, 2012] Florent Teichteil-Königsbuch. Path-Constrained Markov Decision Processes: bridging the gap between probabilistic model-checking and decision-theoretic planning. In *Proc. European Conf. on Artificial Intelligence*, 2012.

[Trevizan and Veloso, 2012] F. W. Trevizan and M. M. Veloso. Short-sighted stochastic shortest path problems. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2012.

[Trevizan *et al.*, 2016] F. W. Trevizan, S. Thiébaux, P. Santana, and B. Williams. Heuristic search in dual space for constrained stochastic shortest path problems. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2016.

[Undurti and How, 2010] Aditya Undurti and Jonathan P. How. An online algorithm for constrained pomdps. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010.