Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Occupation Measure Heuristics for Probabilistic Planning

Felipe Trevizan, Sylvie Thiébaux, Patrik Haslum
Data61, CSIRO and Research School of Computer Science, ANU
Canberra, ACT, Australia
first.last@anu.edu.au

Abstract

For the past 25 years, heuristic search has been used to solve domain-independent probabilistic planning problems, but with heuristics that determinise the problem and ignore precious probabilistic information. To remedy this situation, we explore the use of occupation measures, which represent the expected number of times a given action will be executed in a given state of a policy. By relaxing the well-known linear program that computes them, we derive occupation measure heuristics – the first admissible heuristics for stochastic shortest path problems (SSPs) taking probabilities into account. We show that these heuristics can also be obtained by extending recent operator-counting heuristic formulations used in deterministic planning. Since the heuristics are formulated as linear programs over occupation measures, they can easily be extended to more complex probabilistic planning models, such as constrained SSPs (C-SSPs). Moreover, their formulation can be tightly integrated into i-dual, a recent LP-based heuristic search algorithm for (constrained) SSPs, resulting in a novel probabilistic planning approach in which policy update and heuristic computation work in unison. Our experiments in several domains demonstrate the benefits of these new heuristics and approach.

Introduction

Over the past two decades, heuristic search has established itself as the method of choice for optimal deterministic planning. This is in large part thanks to the strong focus on developing domain-independent admissible heuristics, of which there is now a large supply to choose from – see e.g. works on delete-relaxation (Bonet and Geffner 2001), critical path (Haslum and Geffner 2000), abstraction (Helmert, Haslum, and Hoffmann 2007), landmark (Helmert and Domshlak 2009), operator-counting (van den Briel et al. 2007, Pommerening et al. 2014), and potential heuristics (Pommerening et al. 2015).

Heuristic search also has the potential to be a powerful approach for optimally solving *probabilistic* planning problems, such as stochastic shortest path problems (SSPs), constrained SSPs, and other SSP variants (Mausam and Kolobov 2012). Many search algorithms have been developed for this purpose, including (L)TRDP (Barto, Bradtke, and Singh 1995, Bonet and Geffner 2003), LAO* (Hansen and Zilberstein 2001), FRET (Kolobov et al. 2011, Steinmetz, Hoffmann, and Buffet 2016), and i-dual (Trevizan et al. 2016). However, in contrast to the situation in deterministic planning, the success of these algorithms has been limited by the lack of effective domain-independent heuristics dedicated to the probabilistic planning setting. Existing heuristics simply determinise the problem and fall back on well-established deterministic planning heuristics, failing to exploit valuable information about the probabilities of action outcomes. As far as we are aware, in over two decades of existence of heuristic search algorithms for probabilistic planning, no one has developed admissible heuristics that account for the tradeoff between probabilities and action costs.

To fill this major gap, this paper introduces *occupation measure* heuristics – the first domain-independent admissible heuristics for probabilistic planning that reason about probabilities.[1] An occupation measure is the probabilistic counterpart of an operator count: it represents the expected number of times a given action will be executed in a given state of a policy before the goal is reached. The concept traces back to the *dual linear program* formulation of SSPs (D'Epenoux 1963), which solves SSPs by optimising the policy occupation measures (this contrasts with the more common primal LP formulation where the variables being optimised represent the expected cost to reach the goal). Occupation measure heuristics can therefore be obtained by relaxing the dual LP. We formulate one such relaxation, the projection occupation measure heuristic (h^{pom}), by projecting the dual LP onto individual state variables and enforcing the consistency of the projections' occupation measures. Our experiments show that iLAO* and LRTDP guided by this heuristic often explore significantly fewer nodes than when guided by deterministic planning heuristics.

Similarly to operator-counting heuristics used in the deterministic setting (Pommerening et al. 2014), occupation measure heuristics are formulated as linear programs whose variables are occupation measures. We further relate the two types of heuristics by establishing that h^{roc}, the net-change

[1] Our statement applies to SSPs and not to probabilistic conformant planning or MaxProb type problems for which such heuristics exist, see e.g. (Little, Aberdeen, and Thiébaux 2005, Bryce, Kambhampati, and Smith 2006, Little and Thiébaux 2006, Domshlak and Hoffmann 2007, E.-Martín, Rodríguez-Moreno, and Smith 2014).

heuristic for the all-outcomes determinisation of the SSP, augmented with additional constraints enforcing the respective probabilities of the outcomes of each given operator, fits into the occupation measure heuristic framework and is dominated by h^{pom}. This new heuristic h^{roc} has the merit of requiring substantially fewer LP variables than h^{pom} in typical cases, and results in faster run-times and better scalability than deterministic heuristics in several domains.

One of the strengths of occupation measure heuristics is that they can easily be extended to incorporate additional constraints, such as the bounds on expected costs featured in *constrained* stochastic shortest paths problems (C-SSPs) (Altman 1999, Dolgov and Durfee 2005). In a C-SSP, actions are associated with multiple cost functions (fuel, time, etc), one of which is designated as the primary, and the others as secondary costs, and one seeks a stochastic policy optimising the expected primary cost, subject to bounds on the expected secondary costs. We describe $h^{\text{c-pom}}$ (resp. $h^{\text{c-roc}}$), an extension of h^{pom} (resp. h^{roc}) that incorporates such bounds, and use it to guide i-dual, the state of the art heuristic search algorithm for C-SSPs (Trevizan et al. 2016). We find that $h^{\text{c-pom}}$ and $h^{\text{c-roc}}$ provide stronger guidance, as the heuristics are aware not only of probabilities, but also of the requirements regarding all the secondary costs.

Finally, one of the most intriguing advantages of occupation measure heuristics is that they can be computed at once for multiple states, using the same set of linear constraints. Thus, their formulation can directly be incorporated into the LP solved by i-dual to update the policy at each iteration. This leads to i^2-dual, a brand new type of heuristic search method for C-SSPs where the heuristic computation is lazy, reusable across multiple parts of the search space, and works in unison with the policy update. We find that i^2-dual outperforms i-dual in coverage, time and number of nodes expanded, regardless of the heuristic used by the latter.

To summarise, this paper makes contributions that open up new avenues of research for probabilistic planning: (1) the first heuristics for SSPs and C-SSPs which exploit probabilistic information, (2) a study of their relationship with the operator-counting heuristics used in the deterministic setting, and (3) a new approach to solving C-SSPs which integrates heuristic computation with policy update.

Background: SSPs

We start with some background about stochastic shortest paths problems, which we represent using a probabilistic variant of SAS$^+$. We then follow with a description of relevant solution methods for SSPs, including the dual linear program formulation which optimises occupation measures.

Probabilistic SAS$^+$. A *probabilistic SAS$^+$ task* is a tuple $\langle \mathcal{V}, \mathsf{A}, s_0, s_\star, C \rangle$. $\mathcal{V}$ is a finite set of *state variables*, and each variable v has a finite domain D_v. A *partial state* (or valuation) is a function s on a subset $\mathcal{V}_s$ of $\mathcal{V}$, such that $s[v] \in \mathsf{D}_v$ for $v \in \mathcal{V}_s$ and $v = \bot$ otherwise. If $\mathcal{V}_s = \mathcal{V}$, we say that s is a *state*. s_0 is the *initial state* and $s_\star$ is a partial state representing the *goal*. Given two partial states s and s', we write $s' \subseteq s$ when $s'[v] = s[v]$ for all $v \in \mathcal{V}_{s'}$.

The *result* of applying a (partial) valuation e in state s is the state $res(s,e)$ such that $res(s,e)[v] = e[v]$ if $e[v] \neq \bot$ and $res(s,e)[v] = s[v]$ otherwise. A is a finite set of *probabilistic actions*. Each $a \in \mathsf{A}$ consists of a *precondition* $pre(a)$ represented by a partial valuation over $\mathcal{V}$, a set $\mathit{eff}(a)$ of *effects*, each of which is a partial valuation over $\mathcal{V}$, and a *probability distribution* $\Pr_a(\cdot)$ over effects $e \in \mathit{eff}(a)$ representing the probability of $res(s,e)$ being the state resulting from applying a in s. Finally, $C(a) \in \mathbb{R}^*_+$ is the *immediate cost* of applying a.

Stochastic Shortest Path Problem. A probabilistic SAS$^+$ task is a factored representation of a *Stochastic Shortest Path problem* (SSP) (Bertsekas and Tsitsiklis 1991). A SSP is a tuple $\mathbb{S} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C \rangle$ in which S is the finite set of states, $s_0 \in \mathsf{S}$ is the initial state, $\mathsf{G} \subseteq \mathsf{S}$ is the non-empty set of goal states, A is the finite set of actions, $\mathsf{A}(s)$ is the subset of actions applicable in state s, $P(s'|s,a)$ represents the probability that $s' \in \mathsf{S}$ is reached after applying action $a \in \mathsf{A}(s)$ in state s, and $C(a) \in \mathbb{R}^*_+$ is the immediate cost of applying action a. A solution for the SSP is a *deterministic* stationary *policy* $\pi : \mathsf{S} \mapsto \mathsf{A}$ such that $\pi(s) \in \mathsf{A}(s)$ is the action to be applied in state s. An optimal policy minimises the total expected cost of reaching G from s_0.

Corresponding SSP. The correspondence between SSPs and their probabilistic SAS$^+$ representation is straightforward: a probabilistic SAS$^+$ task $\langle \mathcal{V}, \mathsf{A}, s_0, s_\star, C \rangle$ defines an SSP $\langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C \rangle$ where $\mathsf{S} = \times_{v \in \mathcal{V}} \mathsf{D}_v$, $\mathsf{G} = \{s \in \mathsf{S} | s_\star \subseteq s\}$, $\mathsf{A}(s) = \{a \in \mathsf{A} | pre(a) \subseteq s\}$, and $\Pr(s'|s,a) = \sum_{e \in \mathit{eff}(a) \text{ s.t. } s' = res(s,e)} \Pr_a(e)$.

Dead ends. In this paper, we assume for simplicity that $s_0 \notin \mathsf{G}$ and that the goal is always reachable, i.e., that there are no dead ends.[2] However, our experiments feature problems with dead ends and relax this assumption using the fixed-cost penalty formulation of dead ends (Kolobov, Mausam, and Weld 2012). More principled treatments of dead ends along the lines of (Kolobov, Mausam, and Weld 2012, Teichteil-Königsbuch 2012) are also possible.

Primal Linear Program. It is well-known that SSPs can be solved using linear programming. The most common formulation is the *primal LP formulation* which optimises the policy value function. In this formulation, the variables represent the total expected cost $V(s)$ of reaching the goal from a given state s, and their optimal value V^* is defined by the Bellman equation (1957):

$$V^*(s) = \min_{a \in \mathsf{A}(s)} \sum_{s' \in \mathsf{S}} P(s'|s,a)(C(a) + V^*(s')) \quad (1)$$

for $s \notin \mathsf{G}$ and $V^*(s) = 0$ for $s \in \mathsf{G}$. An optimal policy π^* can be extracted from V^* by replacing min by argmin in the Bellman equation.

Dual Linear Program. Somewhat less well-known in the field of AI is the *dual LP formulation* of SSPs (D'Epenoux 1963, Altman 1999). In this formulation, shown in (LP 1), the variables are the policy's occupation measures $x_{s,a}$ and

[2] This assumption and our strictly positive cost function is equivalent to assuming that there is at least one *proper* policy and that all *improper* policies have infinite cost.

represent the expected number of times action $a \in A(s)$ will be executed in state s.

$$\min_{x} \sum_{s \in \mathsf{S}, a \in \mathsf{A}(s)} x_{s,a} C(a) \quad \text{s.t.} \quad \text{(C1)} - \text{(C6)} \qquad \text{(LP 1)}$$

$$x_{s,a} \geq 0 \qquad \forall s \in \mathsf{S}, a \in \mathsf{A}(s) \quad \text{(C1)}$$

$$out(s_0) - in(s_0) = 1 \quad \text{(C2)}$$

$$\sum_{s_g \in \mathsf{G}} in(s_g) = 1 \quad \text{(C3)}$$

$$out(s) - in(s) = 0 \qquad \forall s \in \mathsf{S} \setminus (\mathsf{G} \cup \{s_0\}) \quad \text{(C4)}$$

$$in(s) = \sum_{s' \in \mathsf{S}, a \in \mathsf{A}(s')} x_{s',a} P(s|s',a) \qquad \forall s \in \mathsf{S} \quad \text{(C5)}$$

$$out(s) = \sum_{a \in \mathsf{A}(s)} x_{s,a} \qquad \forall s \in \mathsf{S} \setminus \mathsf{G} \quad \text{(C6)}$$

This dual formulation can be interpreted as a *probabilistic flow problem*, where $x_{s,a}$ describes the flow leaving state s via action a. The objective function captures the minimisation of the total expected cost to reach the goal (sink) from the initial state (source). Constraints (C2) and (C3) define, respectively, the source (initial state s_0) and the sinks (goal states). For any other state, the flow conservation principle applies, i.e., the flow reaching s must leave s (C4). Finally, constraints (C5) and (C6) define expected flow entering and leaving state s, respectively. The optimal solution x^* of (LP 1) can be converted into an optimal policy $\pi^*(s) = a$ where $a \in \mathsf{A}(s)$ is the only action such that $x^*_{s,a} \neq 0$.

Heuristic Search. Linear programming explores the entire state space at once. In contrast, Heuristic search algorithms for SSPs such as (i)LAO*, LRTDP, and i-dual (Hansen and Zilberstein 2001, Bonet and Geffner 2003, Trevizan et al. 2016) start from the factored problem representation (e.g., as a probabilistic SAS$^+$ task), and incrementally generate parts of the search space, guided by admissible heuristics that estimate the expected cost to reach the goal from each newly generated state (fringe state).

Primal Determinisation Heuristics. Admissible estimates used by these algorithms are typically obtained by relaxing the value function V^* in two steps. Firstly, the problem is determinised: this amounts to replacing the expectation in the Bellman equation (1) with the minimimum over the successor states. This transformation is called the *all-outcomes determinisation* (Jimenez, Coles, and Smith 2006). Secondly, since the resulting deterministic planning problem is still PSPACE-complete, it is further relaxed into an admissible deterministic planning heuristic computable in polynomial time, such as h-max or lm-cut (Bonet and Geffner 2001, Helmert and Domshlak 2009). Both the all-outcomes determinisation and these heuristics are typically computed from the factored problem representation.[3]

[3]In particular, the all-outcomes determinisation of the probabilistic SAS+ task is the deterministic SAS+ task with identical set of variables, initial state, and goal, but whose actions are split into one deterministic action α per probabilistic action $a \in \mathsf{A}$ and effect $e \in eff(a)$, such that $pre(\alpha) = pre(a)$, $eff(\alpha) = \{e\}$, and $C(\alpha) = C(a)$.

Unfortunately, these relaxations of V^* do not take probabilities into account, foregoing valuable information. Yet, in 25 years, it has not been clear how to do better. Whilst it is in principle possible to relax the primal formulation without completely sacrificing probabilities (we do this below), this results in heuristics that are not much more informative than the state of the art, albeit more costly to compute. One of the main contributions of this paper is to achieve informative and efficient heuristics that take probabilities into account by moving from the primal to the dual framework.

Occupation Measure Heuristics for SSPs

Similarly to operator-counting heuristics in deterministic planning, occupation measure heuristics for an SSP $\mathbb{S}$ formalise constraints over real variables $x_{s,a} \geq 0$ for each state $s \in \mathsf{S} \setminus \mathsf{G}$ and action $a \in A(s)$, which must be satisfied by every policy π for $\mathbb{S}$ when setting $x_{s,a}$ to π's occupation measures. The heuristic then optimises the objective of (LP 1) under those constraints.

In this section, we describe one such heuristic, the *Projection Occupation Measure heuristic* h^{pom}, which we obtain by relaxing the dual LP. The key idea is to project the SSP and the dual LP constraints onto individual state variables, and ensure consistency across projections by tying the projection occupation measures together to enforce that the expected number of times a given action is executed is equal in all projections.

More formally, the projection of a probabilistic SAS$^+$ task $\langle \mathcal{V}, \mathsf{A}, s_0, s_\star, C \rangle$ over the state variable $v \in \mathcal{V}$ is the probabilistic SAS$^+$ task in which all states and partial valuations are restricted to the variable v. For this work, we interpret this projection as the SSP $\mathbb{S}^v$ (Definition 1). To ensure we correctly synchronise across projections, this SSP has an extra action a_g leading to an absorbing state g as soon as v reaches its goal value.

Definition 1 (Projection of an SSP). *Given a probabilistic SAS$^+$ task $\langle \mathcal{V}, \mathsf{A}, s_0, s_\star, C \rangle$ and $v \in \mathcal{V}$, its projection from s onto v is the SSP $\mathbb{S}^{v,s} = \langle \mathsf{D}_v \cup \{g\}, s[v], \{g\}, \mathsf{A} \cup \{a_g\}, P, C' \rangle$ where $C'(a_g) = 0$ and $C'(a) = C(a)$ for all $a \in \mathsf{A}$, and*

$$P(d'|d,a) = \begin{cases} \sum_{\substack{e \in eff(a) \text{ s.t.} \\ e[v] = d'}} \Pr_a(e) & \text{if } d \neq d', a \in \mathsf{A}, pre(a)[v] \in \{d, \bot\} \\ \sum_{\substack{e \in eff(a) \text{ s.t.} \\ e[v] \in \{d, \bot\}}} \Pr_a(e) & \text{if } d = d', a \in \mathsf{A}, pre(a)[v] \in \{d, \bot\} \\ 1 & \text{if } d' = g, a = a_g, s_\star[v] \in \{d, \bot\} \\ 0 & \text{otherwise} \end{cases}$$

for all $d \in \mathsf{D}_v$, $d' \in \mathsf{D}_v \cup \{g\}$ and $a \in \mathsf{A} \cup \{a_g\}$. If the state s is omitted, then $s = s_0$.

Given a policy π for $\mathbb{S}$, let the augmented policy π' be $\pi'(s) = a_g$ for all $s \in \mathsf{G}$ and $\pi'(s) = \pi(s)$ otherwise. It is easy to see that π' is executable in any projection of $\mathbb{S}$. However, notice that, while π is stationary over $\mathbb{S}$, π' might be non-stationary over $\mathbb{S}^v$. This is because, a given state $d \in \mathsf{D}_v$ of $\mathbb{S}^v$ might be visited more than once and, at each visit, a different action could be executed depending on the values of the variables $\mathcal{V} \setminus \{v\}$ that are hidden from $\mathbb{S}^v$.

Given $v \in \mathcal{V}$, let $\mathsf{C}^{v,s}$ represent the flow constraints (C1) – (C6) of the dual formulation of $\mathbb{S}^{v,s}$. Each occupation measure of $\mathbb{S}^{v,s}$ is $x^{v,s}_{d,a}$, for $d \in \mathsf{D}_v$ and $a \in \mathsf{A} \cup \{a_g\}$, and represents the expected number of times a is executed in the state d of $\mathbb{S}^{v,s}$. To tie these projection occupation measures and constraints together into a single LP, we add the following tying constraints.

Definition 2 (Tying constraints). *The set of tying constraints for state s, denoted as Tying^s, is*

$$\sum_{d_i \in \mathsf{D}_{v_i}} x^{v_i,s}_{d_i,a} = \sum_{d_j \in \mathsf{D}_{v_j}} x^{v_j,s}_{d_j,a}, \quad \forall\, v_i \in \mathcal{V}, v_j \in \mathcal{V}, a \in \mathsf{A}$$

Any policy that is feasible for the SSP is feasible for all projections and satisfies the tying constraints. These constraints ensure that policies for each projection agree on the expected number of times each action is executed. This synchronisation, however, does not enforce that the actions applied and the states reached at each step need to be consistent across projections. The combination of tying and projection constraints results in the following heuristic.

Definition 3 (Projection occupation measure heuristic). *Given a probabilistic SAS$^+$ task $\langle \mathcal{V}, \mathsf{A}, s_0, s_\star, C\rangle$ the projection occupation measure heuristic h^{pom} at state s is the solution of the following LP:*

$$h^{pom}(s) = \min \sum_{d \in \mathsf{D}_v, a \in \mathsf{A}} x^{v,s}_{d,a} C(a) \;\Big|\; \mathsf{Tying}^s, \mathsf{C}^{v',s} \quad \forall v' \in \mathcal{V},$$

for any variable $v \in \mathcal{V}$.

Notice that, because of the constraints Tying^s, the value of $h^{\text{pom}}(s)$ is the same regardless of which $v \in \mathcal{V}$ is used in the objective function.

Theorem 1 (Admissibility of h^{pom}). *For all states s of the given probabilistic SAS$^+$ task, $h^{pom}(s) \leq V^*(s)$.*

The proof of Theorem 1 can be found in appendix. The proof focuses on the relationship between the optimal occupation measures x^* of $\mathbb{S}$ and the LP defining h^{pom}. In particular, we show that the occupation measures resulting from projecting x^* onto the variables v, satisfy the constraints of this LP. This implies that the objective value h^{pom} of this LP, is less than or equal to the objective value V^* of the dual LP for $\mathbb{S}$.

Note that we could in principle use simpler means to obtain an admissible heuristic estimate taking probabilities into account. We could, for instance, optimally solve (e.g., using the dual or primal LP) each of the projections of $\mathbb{S}$ over state variables whose goal value is defined, and take the maximum of their objective values:

$$h^{\text{pmax}}(s) = \max_{v \in \mathcal{V} \text{ s.t. } s_\star[v] \neq \bot} V^{*,v}(s)$$

where $V^{*,v}(s)$ is the optimal value function for $\mathbb{S}^{v,s}$. However, this estimate is quite loose as it considers that the variables v are independent in $\mathbb{S}$ and its solution corresponds to a deterministic stationary policy for a single projection. In contrast, the solution found by h^{pom} represents a set of stochastic non-stationary policies which are valid for all projections. As our experiments show, the extra representational power of h^{pom} allows it to be a much more informed heuristic. Unlike h^{pmax}, h^{pom} can only be expressed in the dual framework, since the primal formulation lacks the ability to count action occurrences.

Relationship with Operator Counting

Occupation measure heuristics are powerful, but introduce a fair number of LP variables, of the order of $|\mathsf{A}| \times \sum_{v \in \mathcal{V}} |\mathsf{D}_v|$. While this power will be useful for more complex planning tasks dealt with later in the paper, a heuristic not based on projections can obtain similar results to h^{pom} using a different LP with $\sum_{a \in \mathsf{A}} |eff(a)|$ variables which, for most planning tasks, will be much smaller. This new LP exploits the fact that occupation measure heuristics can be seen as the probabilistic counterpart of the operator-counting heuristics introduced in classical deterministic planning, e.g., the net change heuristic (Pommerening et al. 2014). In the deterministic setting, operator-counting heuristics formalise constraints over integer variables $Y_a \geq 0$ for each action a, which must be satisfied by every plan π for the problem when setting Y_a to the number of times a is executed in π. These heuristics optimise $\sum_{a \in \mathsf{A}} Y_a C(a)$ and, for efficiency reasons, consider the LP relaxation of these constraints.

We call our probabilistic version of the operator-counting heuristic the *Regrouped Operator-Counting Heuristic* h^{roc}. The idea behind h^{roc} is to enrich the formulation of the net change heuristic *for the all-outcomes determinisation of the problem*, with constraints that regroup operator counts representing the effects of the same action, and which enforce the relationship between their respective probabilities.

When applied to the all-outcomes determinisation of a given probabilistic SAS$^+$ task, the net change heuristic has a variable $Y_{a,e}$ for each effect e of an action a, which represents the number of times a is executed and e occurs. For each possible state variable assignment (or atom) $v = d \in \mathsf{D}_v$, this heuristic distinguishes between 4 disjoint classes of action/effect pairs, depending on whether they *always produce* (AP), *sometimes produce* (SP), *always consume* (AC) or *sometimes consume* (SC) the atom:

- $AP_{v=d} = \{(a,e) \mid e[v] = d, pre(a)[v] = d' \neq d\}$
- $SP_{v=d} = \{(a,e) \mid e[v] = d, pre(a)[v] = \bot\}$
- $AC_{v=d} = \{(a,e) \mid e[v] = d' \neq d, pre(a)[v] = d\}$
- $SC_{v=d} = \{(a,e) \mid e[v] = d' \neq d, pre(a)[v] = \bot\}$

The heuristic is called "net change" in reference to the change of truth value of an atom from a state to another, where a change of 1 means that the atom becomes true, 0 that it is unchanged, and -1 that it becomes false. The possible net change that a variable can accumulate from a state s where $s[v] = d$ to the goal $s_\star$ is:

$$pnc^{s \to s_\star}_{v=d} = \begin{cases} \{0,1\} & \text{if } s_\star[v] = \bot \text{ and } s[v] \neq d \\ \{-1,0\} & \text{if } s_\star[v] = \bot \text{ and } s[v] = d \\ \{1\} & \text{if } s_\star[v] = d \text{ and } s[v] \neq d \\ \{-1\} & \text{if } s_\star[v] = d' \text{ and } s[v] = d \neq d' \\ \{0\} & \text{otherwise} \end{cases}$$

With these notations, given $v \in \mathcal{V}, d \in \mathsf{D}_v$, and a state s, the net change constraints $\mathsf{N}^{v,d,s}$ are:

$$\sum_{(a,e) \in AP_{v=d}} Y_{a,e} - \sum_{(a,e) \in AC_{v=d}} Y_{a,e} + \sum_{(a,e) \in SP_{v=d}} Y_{a,e} \geq \min pnc^{s \to s_\star}_{v=d} \quad \text{(C7)}$$

$$\sum_{(a,e) \in AP_{v=d}} Y_{a,e} - \sum_{(a,e) \in AC_{v=d}} Y_{a,e} - \sum_{(a,e) \in SC_{v=d}} Y_{a,e} \leq \max pnc^{s \to s_\star}_{v=d} \quad \text{(C8)}$$

In order to recover the information about the probabilistic effects of each action lost by the all-outcomes determinisation (a necessary step to compute $\mathsf{N}^{v,d,s}$), our heuristic h^{roc} uses the following set of constraints:

Definition 4 (Regrouping constraints). *The set of regrouping constraints, denoted as* Regroup*, is*

$$\Pr_a(e_1)Y_{a,e_2} = \Pr_a(e_2)Y_{a,e_1} \quad \forall\, a \in \mathsf{A}, \{e_1, e_2\} \in eff(a).$$

These constraints enforce that the expected number of times outcome e_1 of action a occurs is proportional with a factor $\Pr_a(e_1)/\Pr_a(e_2)$ to the expected number of times any other outcome e_2 of the same action occurs. Therefore, not only the probability of each effect is recovered, but also the effect dependency i.e., $e_1 > 0$ implies $e_i > 0$ for all $e_i \in eff(a)$.

The heuristic h^{roc} is presented in Definition 5. Theorem 2 shows that h^{pom} dominates h^{roc}; therefore h^{roc} is admissible.

Definition 5 (Regrouped operator-counting heuristic). *Given a probabilistic SAS$^+$ task, the regrouped operator-counting heuristic h^{roc} at state s is the solution of the LP:*

$$h^{roc}(s) = \min_{Y} \sum_{a,e} Y_{a,e} C(a) \mid \mathsf{Regroup}, \mathsf{N}^{v,d,s}\ \forall v \in \mathcal{V}, d \in \mathsf{D}_v$$

Theorem 2 (h^{pom} dominates h^{roc}). *For all state s of the given probabilistic SAS$^+$ task, $h^{roc}(s) \leq h^{pom}(s)$.*

The proof of Theorem 2 is in the appendix and it consists in constructing a feasible solution for the LP solved by h^{roc} based on the optimal solution of the LP solved by h^{pom} and showing that both solutions have the same cost.

Similarly to the operator-counting heuristics (including h^{roc}), our projection occupation measure heuristic can also be augmented with constraints that represent other state-of-the-art heuristics, e.g., *disjunctive action landmarks* (Pommerening et al. 2014). This transformation of operator-counting constraints to projection occupation measure constraints is formalized by Corollary 3 of Theorem 2.

Corollary 3. *Any operator-counting constraint over the variables $Y_{a,e}$ for h^{roc} can be translated to a constraint for h^{pom} by replacing $Y_{a,e}$ with $\Pr_a(e) \sum_{d \in \mathsf{D}_v} x^{v,s}_{d,a}$.*

Proof. By the regrouping constraints, $Y_{a,e'}$ equals $Y_{a,e} \Pr_a(e')/\Pr_a(e)$ thus $\sum_{e' \in eff(a)} Y_{a,e'} = Y_{a,e}/\Pr_a(e)$ for all $e \in eff(a)$. Moreover, $\sum_{e' \in eff(a)} Y_{a,e'}$ is the expected number of times that action a is executed and it is equivalent to $\sum_{d \in \mathsf{D}_v} x^{v,s}_{d,a}$ for h^{pom} for any $v \in \mathcal{V}$. □

More Background: C-SSPs

One of the strengths of occupation measure heuristics is that they are well-suited to solving more complex probabilistic planning problems allowing objectives and additional constraints that can be formulated in terms of occupation measures. In the rest of the paper, we extend occupation measure heuristics to Constrained SSPs (C-SSPs) (Altman 1999), which are a general model for planning uncertainty under multiple competing objectives. These objectives are captured by multiple cost functions, one of which is optimised while constraining the others. For example, a C-SSP allows the minimisation of the policy's expected fuel consumption while keeping the expected time to the goal and the risk of failure below acceptable thresholds.

Constrained SSPs and Probabilistic SAS$^+$ tasks. A C-SSP $\mathbb{C} = \langle \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, \vec{C}, \vec{u} \rangle$ is an SSP whose cost function is replaced by a vector of $n+1$ cost functions $\vec{C} = [C_0, \ldots, C_n]$ ($C_j \colon \mathsf{A} \to \mathbb{R}^*_+$ for all j) and a vector of n bounds $\vec{u} = [u_1, \ldots, u_n]$ ($u_j > 0$ for all j). We refer to C_0 as the *primary cost* and to the other elements of the cost vector as the *secondary costs*. An optimal solution for a C-SSP is a stochastic policy $\pi : \mathsf{S} \mapsto \mathsf{A} \times [0, 1]$, which minimises the expected primary cost C_0 to reach the goal G from the initial state s_0, subject to the expected values of the secondary cost C_j being upper bounded by u_j for $j \in \{1, \ldots, n\}$. Whereas for SSPs there always exists an optimal deterministic policy, stochastic policies are needed to optimally account for trade-offs between the various cost functions. Nevertheless, the complexity of optimally solving C-SSPs remains polynomial in the size of the C-SSP (Dolgov and Durfee 2005). Naturally, a C-SSP can be compactly represented by a *constrained probabilistic SAS$^+$ task*, i.e. a probabilistic SAS$^+$ task whose cost function has been replaced with the corresponding vectors of cost functions and upper bounds.

Dual LP formulation of C-SSPs. From the definition of C-SSPs, it follows that they can be solved by the dual LP formulation of SSPs (LP 1), by replacing C with C_0 in the objective function and adding the following constraint (C9):

$$\sum_{s \in \mathsf{S}, a \in \mathsf{A}(s)} x_{s,a} C_j(a) \leq u_j \qquad \forall j \in \{1, \ldots, n\} \qquad \text{(C9)}$$

Note that attempting to encode these constraints into the primal LP would lead to a nonlinear program involving bilinear constraints. In contrast, the dual program for C-SSP remains linear, but unlike in the SSP case, returns a potentially stochastic policy given by $\pi^*(a, s) = x^*_{s,a}/out(s)$.

Heuristic Search for C-SSPs. The main computational burden with the dual LP is that it requires encoding and exploring all states reachable from s_0. I-dual is a heuristic search algorithm for C-SSPs which alleviates this issue (Trevizan et al. 2016). It explores incrementally larger *partial problems* starting from s_0, using a set of artificial goal states $\hat{\mathsf{G}}$ to represent unexplored areas of the occupation measure space. When first reached, these artificial goal states incur *terminal costs* given by a vector $\vec{H} = [H_0, \ldots, H_n]$ of admissible heuristic functions, where H_j underestimates the expected cost C_j of reaching G. At each iteration, i-dual expands the fringe states F_R that are *reachable* under current best policy. This leads to a new partial problem, i.e. a C-SSP with terminal costs $\hat{\mathbb{C}} = \langle \hat{\mathsf{S}}, s_0, \hat{\mathsf{G}}, \hat{\mathsf{A}}, P, \vec{C}, \vec{u}, \vec{H} \rangle$, where $\hat{\mathsf{G}} = \mathsf{F} \cup (\mathsf{G} \cap \hat{\mathsf{S}})$, i.e., the union of all fringe states (F) and goals seen so far. The current best policy is updated by solving $\hat{\mathbb{C}}$ using the dual LP formulation of C-SSPs, slightly extended to account for terminal costs:

$$\min_x \sum_{s \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s)} x_{s,a} C_0(s, a) + \sum_{s_g \in \hat{\mathsf{G}}} in(s_g) H_0(s_g)$$
$$\text{s.t.} \quad \text{(C1)} - \text{(C6)}, \text{(C9)} - \text{(C10)} \qquad \text{(LP 2)}$$

$$\sum_{s \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s)} x_{s,a} C_j(s, a) + \sum_{s_g \in \hat{\mathsf{G}}} in(s_g) H_j(s_g) \leq u_j \qquad \forall j \in \{1, \ldots, n\} \quad \text{(C10)}$$

I-dual terminates when all fringe states reachable under the current best policy are goal states of the original C-SSP $\mathbb{C}$, i.e., when $\mathsf{F}_R \subseteq \mathsf{G}$. If all heuristics are admissible, the resulting policy is the optimal stochastic policy for $\mathbb{C}$.

Trevizan et al. (2016) tested i-dual with primal determinisation heuristics H_j (such as h-max or lm-cut) for a relaxation of $\mathbb{C}$ that ignores the constraints and optimises C_j. That is, $H_j(s)$ is an admissible heuristic for the regular SSP $\langle \mathsf{S}, s, \mathsf{G}, \mathsf{A}, P, C_j \rangle$. Unfortunately, such individual heuristic H_j have low accuracy: not only they assume that probabilities are irrelevant, but also that the various cost functions do not interact. For instance a heuristic estimating expected fuel consumption may believe that very little fuel is needed because it completely disregards constraints on expected travel time. As we show below, occupation measure heuristics enable us to remedy both issues.

Occupation measures Heuristics for C-SSPs

Since bounds on expected secondary costs can be expressed by means of linear constraints over occupation measures, extending occupation measures heuristics to include these bounds is straightforward. The resulting heuristics account for probabilities and for the dependence between cost functions, and are suitable for constrained probabilistic SAS$^+$ tasks representing C-SSPs. Below we define such a heuristic, $h^{\text{c-pom}}$, which extends h^{pom} to constrained problems. We also define a constrained formulation of h^{roc}, and prove the admissibility of the two heuristics.

Definition 6 (Constrained projection occupation measure and regrouped operator-counting heuristics). *Given a constrained probabilistic SAS$^+$ task $\langle \mathcal{V}, \mathsf{A}, s_0, s_\star, \vec{C}, \vec{u} \rangle$, the constrained projection occupation measure heuristic $h^{c\text{-}pom}$ at state s is the solution of the following LPs:*

$$h^{c\text{-}pom}(s) = \min \sum_{d \in \mathsf{D}_v, a \in \mathsf{A}} x^{v,s}_{d,a} C_0(a) \;\Big|\; \mathsf{CostUB}, \mathsf{Tying}^s, \mathsf{C}^{v',s} \; \forall v' \in \mathcal{V}$$

for any variable $v \in \mathcal{V}$, where CostUB is the constraint set:

$$\sum_{d \in \mathsf{D}_v, a \in \mathsf{A}} x^{v,s}_{d,a} C_j(a) \leq u_j \quad \forall j \in \{1, \ldots, n\}.$$

The constrained regrouped operator-counting heuristic $h^{c\text{-}roc}$ at state s is the solution of the following LP:

$$h^{c\text{-}roc}(s) = \min \sum_{a \in \mathsf{A}, e \in eff(a)} Y_{a,e} C_0(a)$$
$$\text{s.t. } \mathsf{CostUB}', \mathsf{Regroup}, \mathsf{N}^{v,d,s} \; \forall v \in \mathcal{V}, d \in \mathsf{D}_v$$

where CostUB' is the constraint set:

$$\sum_{a \in \mathsf{A} e \in eff(a)} Y_{a,e} C_j(a) \leq u_j \quad \forall j \in \{1, \ldots, n\}$$

Theorem 4 (Admissibility of $h^{\text{c-pom}}$ and $h^{\text{c-roc}}$; dominance of $h^{\text{c-pom}}$). *For all states s of the given constrained probabilistic SAS$^+$ task, $h^{c\text{-}roc}(s) \leq h^{c\text{-}pom}(s) \leq V^*(s)$.*

The admissibility proof follows from the admissibility of h^{pom} (Theorem 1) and h^{roc} (Theorem 2), and the fact that, for all $a \in \mathsf{A}$, $\sum_{d \in \mathsf{D}_v} x^{v,s}_{d,a}$ and $\sum_{a \in \mathsf{A}, e \in eff(a)} Y_{a,e}$ are both lower bounds on the expected number of times action a is executed in state s. The dominance proof follows from the dominance of h^{pom} (Theorem 2) and from Corollary 3.

Using these heuristics in conjunction with i-dual is straightforward: we call i-dual with the heuristic vector $\vec{H}$ such that H_j is $h^{\text{c-pom}}$ or $h^{\text{c-roc}}$ for the constrained SAS$^+$ probabilistic task $\langle \mathcal{V}, \mathsf{A}, s_0, s_\star, [C_j, C_1, \ldots, C_n], \vec{u} \rangle$. That is, to compute the heuristic for a given cost function C_j, we substitute C_j for the primary cost C_0 of the problem in Definition 6. As our experiments show, i-dual equipped with such $\vec{H}$ often explores substantially fewer states to find an optimal stochastic policy than with primal determinisation heuristics.

Heuristics $h^{\text{c-pom}}$ and $h^{\text{c-roc}}$ account for the constraints in an admissible way, but use the cost bounds u_j regardless of whether the artificial goal state s is s_0 or is reached far down the policy. In principle, we would like to make these heuristics tighter by keeping track of the expected costs g_j incurred before reaching an artificial goal state of the policy under consideration (analogously to the function g of A*), and using $u_j - g_j$ as the bounds in place of u_j in the CostUB and CostUB' constraints. However, this seems at first glance impossible to do, since g_j is policy dependent, and the policy update step of i-dual (in which the heuristics we are seeking to compute are used) explores the entire (infinite) stochastic policy space for the current partial problem at once.

Our final contribution, in the next section, is i^2-dual, a variant of i-dual which achieves this by integrating the computation of the $h^{\text{c-pom}}$ heuristic and the policy update into a single LP. This LP explores the policy space while simultaneously performing *lazy* heuristic computation. It also reuses parts of heuristic computations corresponding to different projections across the state and policy space. As far as we are aware, this constitutes a significant departure from existing approaches in the literature.

Heuristic Computation Within Policy Update

In a nutshell, our LP integrating the heuristic computation within policy update is the union of the dual LP solved by i-dual (LP 2) and the LP solved to compute $h^{\text{c-pom}}$ (Definition 6) at the reachable fringe states – albeit with the tighter cost upper bounds. Since the reachable fringe states and their probability of being reached are dependent on the policy being computed, the key challenge is to link these two LPs by passing the correct probability flow to the $h^{\text{c-pom}}$ computation, without explicit reference to each individual reachable fringe state. We achieve this as follows.

Firstly, we generalise the set of flow constraints $\mathsf{C}^{v,s}$ for the projections onto v to not depend on s for initial state, but instead, to use a probability distribution p^v_0 over initial states. Formally, given $v \in \mathcal{V}$, let p^v_0 be a probability distribution over $\mathsf{D}_v \cup \{g\}$, then the flow constraints C^{v,p^v_0} represents the dual formulation constraints for $\mathbb{S}^{v,p^v_0}$ (i.e., the projected SSP with probabilistic initial state) and is defined as:

$$x^v_{d,a} \geq 0 \qquad \forall d \in \mathsf{D}_v, a \in \mathsf{A} \quad \text{(C11)}$$

$$out^v(d) - in^v(d) = p^v_0(d) \qquad \forall d \in \mathsf{D}_v \quad \text{(C12)}$$

$$in^v(g) = 1 \quad \text{(C13)}$$

$$in^v(d) = \sum_{d' \in \mathsf{D}_v, a \in \mathsf{A} \cup \{a_g\}} x^v_{d',a} P(d|d',a) \qquad \forall d \in \mathsf{D}_v \cup \{g\} \quad \text{(C14)}$$

$$out^v(d) = \sum_{a \in \mathsf{A} \cup \{a_g\}} x^v_{d,a} \qquad \forall d \in \mathsf{D}_v \quad \text{(C15)}$$

As expected, $\mathsf{C}^{v,s}$ is the special case of C^{v,p^v_0} for $p^v_0(s[v]) = 1$. Moreover, the probabilistic initial state p^v_0 allows us to redirect towards the heuristic computation the total probability mass of all reachable fringe states, including the goal probability mass via $p^v_0(g)$. This leads to the following combined LP, where as before, $v \in \mathcal{V}$ is any state variable:

$$\min_x \sum_{s \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s)} x_{s,a} C_0(a) + \sum_{d \in \mathsf{D}_v, a \in \mathsf{A}} x^v_{d,a} C_0(a) \quad \text{s.t.} \quad \text{(C16) – (C25)} \qquad \text{(LP 3)}$$

$$x_{s,a} \geq 0 \qquad \forall s \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s) \quad \text{(C16)}$$

$$out(s_0) - in(s_0) = 1 \quad \text{(C17)}$$

$$out(s) - in(s) = 0 \qquad \forall s \in \hat{\mathsf{S}} \setminus \hat{\mathsf{G}} \quad \text{(C18)}$$

$$in(s) = \sum_{s' \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s')} x_{s',a} P(s|s',a) \qquad \forall s \in \hat{\mathsf{S}} \quad \text{(C19)}$$

$$out(s) = \sum_{a \in \hat{\mathsf{A}}(s)} x_{s,a} \qquad \forall s \in \hat{\mathsf{S}} \setminus \hat{\mathsf{G}} \quad \text{(C20)}$$

$$\sum_{s \in \hat{\mathsf{S}}, a \in \hat{\mathsf{A}}(s)} x_{s,a} C_j(a) + \sum_{d \in \mathsf{D}_v, a \in \mathsf{A}} x^v_{d,a} C_j(a) \leq u_i \qquad \forall j \in \{1, \ldots, n\} \quad \text{(C21)}$$

$$p^{v'}_0(g) = \sum_{s_g \in \hat{\mathsf{S}} \cap \mathsf{G}} in(s_g) \qquad \forall v' \in \mathcal{V} \quad \text{(C22)}$$

$$p^{v'}_0(d) = \sum_{s_f \in \mathsf{F}, s_f[v'] = d} in(s_f) \qquad \forall v' \in \mathcal{V}, d \in \mathsf{D}_{v'} \quad \text{(C23)}$$

$$\sum_{d_i \in \mathsf{D}_{v_i}} x^{v_i}_{d,a} = \sum_{d_j \in \mathsf{D}_{v_j}} x^{v_j}_{d,a} \qquad \forall v_i, v_j \in \mathcal{V}, a \in A \quad \text{(C24)}$$

and constraints $\mathsf{C}^{v', p^{v'}_0} \ \forall v' \in \mathcal{V}$ (C25)

The second sum in the objective function estimates the expected primary costs of reachable fringe states, and is equivalent to $\sum_{s_g \in \hat{G}} in(s_g) H_0(s_g)$ when H_0 is $h^{\text{c-pom}}$ with the tighter secondary cost bounds $u_j - g_j$. (C16) – (C20) represent the dual constraints formulation for a regular SSP with the sink constraint omitted. (C21) are the secondary cost constraints where the second summation is obtained using the computed heuristics, similarly to the objective function. (C22) – (C23) define the probabilistic initial state of each projection as the probability mass of reaching, respectively, the goal of the original problem and fringe states satisfying $v' = d$. (C24) is the set of tying constraints (Definition 2) written using the projection occupation measure variables x^{v_i} and x^{v_j} that don't depend on a state s. (C25) represents each of the projections of the problem onto v' using $p^{v'}_0$ as probabilistic initial state. Lastly, the sink constraint equivalent to (C3) is enforced by (C13) of each projection. Formally, since p^v_0 is a probability distribution, (C13) is equivalent to $\sum_{s_g \in \hat{\mathsf{S}} \cap \mathsf{G}} in(s_g) + \sum_{s_f \in \mathsf{F}} in(s_f) = 1$ by (C22) and (C23) for each $\mathsf{C}^{v', p^{v'}_0}$.

The key insight for this integrated version i^2-dual, is that p^v_0, for each v, is not fixed, instead, it's a set of free variables that the LP solver is optimizing. Moreover, p^v_0 bridges two LPs: the LP solving the current C-SSP and the LP computing the heuristics; therefore, a change in any of the variables in one of these LPs is propagated to the other. The result is a completely new approach where policy update and heuristic computation work in unison, without one driving the other.

Note that it is not possible to integrate $h^{\text{c-roc}}$ with i-dual while remaining in the LP framework. This is because operator-counting variables only represent actions (and their outcomes) whereas the occupation measure variables also represent the state. It is this feature that enables occupation measure formulations to compute the heuristic for multiple states at once using the same set of constraints. Formally, an integration of $h^{\text{c-roc}}$ to i-dual requires s to be a free variable to be optimised; this in turn means that $pmc^{s \to s_\star}_{v=d}$ can no longer be a constant and that integer variables must be introduced to capture the case statements in its definition.

Empirical Evaluation

In this section we empirically evaluate our new heuristics for SSPs and C-SSPs, and our new planner i^2-dual. All our results represent the average over 30 runs of each combination of planner and heuristic. We enforce a 30-minutes and 4-Gb cut-off for all experiments. Due to space limitations, a comprehensive description of domains and the full table of results for each domain is in the appendix.

Stochastic Shortest Path Problems

For SSPs, we compare our new heuristics h^{roc} and h^{pom} against (i) the determinisation-based heuristics $h^{\max}$, h^{lmc} and net change heuristic h^{net}, and (ii) the trivial max-projection heuristic h^{pmax}. We use LRTDP and iLAO* as the search algorithms for this comparison. For completeness, the appendix also reports results when i-dual and i^2-dual are used, but we do not consider them further in this subsection since they are not designed for ordinary SSPs and perform poorly as expected. We consider the following domains:

Blocks World (IPPC'08). Extension of the well-known deterministic blocks world domain in which the actions pick-up and put-on-block might fail with probability 0.25. Moreover, three new probabilistic actions allow towers of two blocks to be manipulated: pick-tower, put-tower-on-block, and put-tower-down.

Exploding blocks world (IPPC'08). Extension of the deterministic blocks world domain in which blocks can explode and destroy other blocks or the table. Once a block or the table is destroyed, nothing can be placed on them, and

		LRTDP					iLAO*				
		$h^{\max}$	h^{lmc}	h^{net}	h^{roc}	h^{pom}	$h^{\max}$	h^{lmc}	h^{net}	h^{roc}	h^{pom}
Blocks World	8	3	0	26	30	30	2	30	30	**30**	30
	8	28	0	30	30	30	30	30	30	**30**	30
	8	2	0	12	30	29	2	30	30	**30**	30
	10	0	0	0	30	18	0	0	1	**30**	30
	10	0	0	0	30	0	0	0	0	**30**	30
	12	0	0	0	0	0	0	0	0	**30**	5
Parc Printer	F,4,2	30	30	30	**30**	30	4	30	30	30	30
	F,4,3	30	30	30	**30**	30	0	30	30	30	30
	F,5,2	0	30	0	**30**	0	2	16	0	30	0
	F,5,3	0	30	0	**30**	0	0	0	0	30	0
	T,4,2	0	0	0	1	0	1	30	30	**30**	0
	T,4,3	0	0	0	0	0	0	30	30	**30**	0
	T,5,1	0	0	0	0	0	0	0	0	**30**	0
Exploding BW	7	30	30	30	30	30	30	**30**	30	30	30
	8	30	30	0	**30**	0	0	0	0	3	0
	9	**30**	30	0	**30**	30	30	30	0	30	30
	10	**30**	30	0	**30**	0	23	4	0	11	1
	11	0	0	0	0	0	12	6	0	**16**	0
	12	0	0	0	0	0	24	15	0	**26**	0
	15	0	0	0	0	0	**28**	12	0	23	0
Triag. Tire	3	**30**	30	30	30	30	30	30	30	30	30
	4	**30**	30	30	30	30	30	30	30	30	30
	5	**30**	24	0	30	0	0	0	0	4	0
	6	0	0	0	**30**	0	0	0	0	0	0

Table 1: Coverage for selected SSP problems. Best planner (i.e., fastest planner to obtain the best coverage) in bold. Dead-end variant of the h^{roc} and h^{pom} used in the gray cells. Parameters: number of blocks for blocks world; (has repair action,s,c) for parc printer; and IPPC'08 problem number for exploding blocks world and triangle tire world.

destroyed blocks cannot be moved; therefore, problems in this domain can have unavoidable dead ends.

Triangle Tire World (IPPC'08). This domain represents a car that has to travel between locations in order to reach a goal location from its initial location. When the car moves between locations, a flat tire happens with probability 0.5 and the car becomes unable to move if both the car and the location do not have a spare tire. Problems in this domain are generated to have avoidable dead ends.

Probabilistic Parc Printer. Probabilistic extension of the sequential Parc Printer domain from IPC in which s sheets need to be printed on a modular printer. The printer has c unreliable components in which a sheet can jam with probability 0.1 making the component unavailable and requiring a new exemplar of this sheet to be printed. The unavailability of components creates avoidable dead ends. Also, a high-cost repair action that removes all jams and restores availability of all components can be available.

Table 1 presents coverage results for a subset of the problems solved and the following is a summary of our findings from the experiments in appendix:

Does taking probability into account in the heuristic help? To answer this question, we compare the performance of h^{net} against h^{roc} since the only difference between them is that h^{roc} takes probability into account through the regrouping constraints. For blocks world, tire world and parc printer, planners using h^{roc} obtained a speed up w.r.t. to h^{net} between 2x-56x, 1.3x-10x, and 1.1x-14x, respectively. Moreover, planners using h^{roc} were able to scale up to larger problems than when using h^{net}: 10 blocks vs 8 blocks for blocks world, 5 vs 4 sheets for parc printer, and problem #5 vs #4 for tire world. For exploding blocks world, there was no statistically significant difference between h^{roc} and h^{net}.

Is h^{pom} better than h^{roc}? No. For all the problems considered, a planner using h^{roc} outperforms the same planner using h^{pom} in both runtime and scalability. Moreover, this difference is statistically relevant, specially for the runtime: planners using h^{roc} are up to 25x, 8x, 46x and 34x faster than the same planner using h^{pom} for blocks world, tire world, parc printer and exploding blocks world, respectively. This runtime difference is because h^{pom} and h^{roc} returned the same heuristic values for the considered problems and the LPs solved by h^{pom} have considerably more variables than the LPs solved by h^{roc}. The difference between the number of states explored by a planner using h^{pom} and h^{roc} is statistically insignificant which also illustrate this point.

How do h^{pom} and h^{roc} compare to the state-of-the-art? For blocks world, planners using h^{pom} and h^{roc} are the only ones that scale up to problems with 10 blocks and the best performance overall is obtained by iLAO* with h^{roc}. For parc printer, h^{roc} outperforms all other heuristics and h^{lmc} outperforms h^{pom} for the planners considered. The best performance in this domain alternates between LRTDP with h^{roc} and iLAO* with h^{roc}. For tire world LRTDP with $h^{\max}$ is the best planner closely followed by LRTDP with h^{roc} as the problem size increases up to problem #5. LRTDP with h^{roc} is the only planner that can handle problem #6. A similar trend happens for iLAO* with $h^{\max}$ and h^{roc}. For i-dual, h^{roc} is always better than $h^{\max}$.

Except in exploding blocks world, h^{pom} and h^{roc} expand much fewer states, e.g., up to 48x less than $h^{\max}$ and 10x less than h^{net} in parc printer, 5x times less than h^{lmc} in blocks world. For exploding blocks world, planners using h^{net}, h^{roc} and h^{pom} perform poorly as they do not detect dead ends as early as $h^{\max}$ and h^{lmc}. This advantage of $h^{\max}$ and h^{lmc} is due to two reasons: (i) a state s has zero probability of reaching the goal iff it is a dead end in the all-outcomes determinisation, thus $h^{\max}$ and h^{lmc} are aware of dead ends even though they ignore probabilities; and (ii) for this domain, the dead ends are reached when a precondition of an action that potentially leads to the goal becomes false, thus $h^{\max}$ and h^{lmc} can easily find the dead ends since they propagate the actions preconditions. To illustrate these points, we augmented h^{roc} and h^{pom} with $h^{\max}$ as a dead-end detector. Formally, $h^{\text{roc}}_{\text{de}}(s)$ equals the dead-end penalty if $h^{\max}$ reports that s is a dead end and $h^{\text{roc}}(s)$ otherwise (similarly for $h^{\text{pom}}_{\text{de}}$). The results for $h^{\text{roc}}_{\text{de}}$ and $h^{\text{pom}}_{\text{de}}$ corroborate the above explanation because of the large increase in performance when compared against h^{roc} and h^{pom}, respectively. Moreover, planners using $h^{\max}$ and $h^{\text{roc}}_{\text{de}}$ perform similarly and the best heuristic for a given problem alternates between them: $h^{\max}$ is better in 4 problems, $h^{\text{roc}}_{\text{de}}$ is better in 3 problems, and

		i-dual						i^2-dual
		$h^{\max}$	$h^{\text{lmc-m}}$	h^{roc}	$h^{\text{c-roc}}$	h^{pom}	$h^{\text{c-pom}}$	
Search and Rescue	$4, 0.50, 3$	**30**	**30**	**30**	**30**	30	30	**30**
	$4, 0.50, 4$	29	**30**	30	30	29	30	**30**
	$4, 0.75, 3$	26	**30**	29	29	28	28	**30**
	$4, 0.75, 4$	0	4	1	1	1	1	**7**
	$5, 0.50, 3$	30	**30**	30	30	30	30	**30**
	$5, 0.50, 4$	5	9	9	9	9	9	**14**
	$5, 0.75, 3$	19	**28**	23	23	20	21	**28**
	$5, 0.75, 4$	0	2	2	2	1	1	**6**
Parc Printer	$0,\ 1$	30	30	30	30	25	28	**30**
	$0, \infty$	30	**30**	30	30	30	30	**30**
	$0.1,\ 1$	0	0	0	30	0	27	**30**
	$0.1, \infty$	0	0	0	30	0	30	**30**
	$0.2,\ 1$	0	0	0	0	0	0	**30**
	$0.2, \infty$	0	0	0	0	0	0	**30**

Table 2: Coverage for selected C-SSP problems. Best planner (i.e., fastest planner to obtain the best coverage) in bold. For the parc printer problems shown, $s = 4$, $c = 3$ and no repair action is available. Parameters: (n, r, d) for search and rescue; and (f, u) upper bounds for parc printer.

the difference is statistically insignificant for 2 problems.

Constrained SSPs

For C-SSPs, we compare i^2-dual against i-dual using as heuristic vector $\vec{H} = [h, \ldots, h]$ for h equal to: (i) the SSP heuristics $h^{\max}$, h^{roc} and h^{pom}; and (ii) the cost-constrained heuristics $h^{\text{c-roc}}$ and $h^{\text{c-pom}}$. We also consider the heuristic vector $[h^{\text{lmc}}, h^{\max}, \ldots, h^{\max}]$, i.e., h^{lmc} for C_0 and $h^{\max}$ for the other cost functions; we refer to this heuristic vector as $h^{\text{lmc-m}}$. The planners are evaluated in two domains:

Search and rescue. Domain from (Trevizan et al. 2016) in which a robot has to navigate an $n \times n$ grid and its goal is to find a survivor 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 another survivor. Thus, the planner has to trade off fuel, exploration of unknown survivors, and time to rescue. The other parameters of a problem are: d, the distance to the known survivor; and r, the density of potential survivors.

Constrained Parc Printer. Extension of the probabilistic parc printer domain in which the expected number of jams is upper-bounded by f. Also, the expected number of times the reliable finisher component can be used is upper-bounded by u. When this threshold is reached, only a more expensive and potentially unreliable finisher component is available.

Table 2 presents coverage results for a subset of the problems solved and the following is a summary of our findings from the experiments in appendix:

Does the constraints in the heuristics help? Yes. Comparing h^{roc} against $h^{\text{c-roc}}$ and h^{pom} against $h^{\text{c-pom}}$, we observed a small improvement in coverage and no statistically relevant speed up for the search and rescue domain. For the parc printer domain, the improvement in coverage is significant: $h^{\text{c-pom}}$ obtained at least 63% in 22 problems in which h^{pom} has 0% coverage; and $h^{\text{c-roc}}$ obtained 100% coverage in 16 problems in which h^{roc} has 0% coverage.

Is i^2-dual better than i-dual using $h^{\text{c-pom}}$? Yes and, in both domains, i^2-dual is the best overall planner. For the search and rescue domain, i^2-dual obtained the best coverage in all the problems, tying with $h^{\text{lmc-m}}$ in the small and medium problems at 100% and solving up to 3 times more instances than $h^{\text{lmc-m}}$ for the larger problems. Regarding runtime, there is no statistically significant difference between i^2-dual and $h^{\text{lmc-m}}$ for the problems in which both obtained the same coverage. For the constrained parc printer domain, i^2-dual outperforms all other planners in both coverage and runtime: it obtained coverage between 30% and 100% in 13 problems that no other planner was able to solve and up to 34x speed up w.r.t. the second best planner.

Conclusion

In this paper, we have presented what we believe to be the first domain-independent admissible heuristics specifically designed to exploit the interactions between probabilities and action costs found in SSPs and C-SSPs. We have shown that they perform well across a range of domains and search algorithms, and that handling probabilities in heuristics often pays. Previous heuristics exploiting outcome probabilities have only considered MaxProb type problems, and used the planning graph data structure which can yield poor estimates when policies are cyclic (Little and Thiébaux 2006). One area of future work is to improve the accuracy of our heuristics by augmenting their formulation with merges and disjunctive action landmarks (and other operator counting constraints), as was done in the deterministic setting by Bonet and van den Briel (2014).

We have established a bridge between the more general occupation measure constraints and the operator counting constraints used in deterministic planning (Pommerening et al. 2014). Future work should settle the question of whether the h^{pom} and h^{roc} heuristics are equivalent. We have not, so far, found a single counter-example to this, but have only managed to prove equivalence in the absence of "sometimes consumers/producers". In the deterministic setting, Pommerening et al. (2015) have established the equivalence of the net change heuristics and projection heuristics under optimal general cost partitioning. However, it is not obvious to us how to adapt their proof to our setting where optimal cost partitioning is replaced with tying constraints.

Finally, we have introduced i^2-dual, a new state of the art method for C-SSPs in which policy update and heuristic computation are fully synergistic. We believe that the principles behind i^2-dual can be replicated to incorporate path-dependent heuristics into other algorithms.

Acknowledgements

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

References

Altman, E. 1999. *Constrained Markov Decision Processes*, volume 7. CRC Press.

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *Artif. Intell.* 72(1-2):81–138.

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

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

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artif. Intell.* 129(1-2):5–33.

Bonet, B., and Geffner, H. 2003. Labeled RTDP: improving the convergence of real-time dynamic programming. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Bonet, B., and van den Briel, M. 2014. Flow-based heuristics for optimal planning: Landmarks and merges. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Bryce, D.; Kambhampati, S.; and Smith, D. E. 2006. Sequential monte carlo in probabilistic planning reachability heuristics. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 233–242.

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

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

Domshlak, C., and Hoffmann, J. 2007. Probabilistic planning via heuristic forward search and weighted model counting. *J. Artif. Intell. Res. (JAIR)* 30:565–620.

E.-Martín, Y.; Rodríguez-Moreno, M. D.; and Smith, D. E. 2014. Progressive heuristic search for probabilistic planning based on interaction estimates. *Expert Systems* 31(5):421–436.

Hansen, E. A., and Zilberstein, S. 2001. LAO: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* 129(1):35–62.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proc. Int. Conf. of Artificial Intelligence Planning Systems*, 140–149.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 176–183.

Jimenez, S.; Coles, A.; and Smith, A. 2006. Planning in probabilistic domains using a deterministic numeric planner. In *Proc. Workshop of the UK Planning and Scheduling Special Interest Group*.

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic search for generalized stochastic shortest path mdps. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Kolobov, A.; Mausam; and Weld, D. S. 2012. A theory of goal-oriented mdps with dead ends. In *Proc. Conf. on Uncertainty in Artificial Intelligence (UAI)*.

Little, I.; Aberdeen, D.; and Thiébaux, S. 2005. Prottle: A probabilistic temporal planner. In *Proc. of National Conference on Artificial Intelligence (AAAI)*, 1181–1186.

Little, I., and Thiébaux, S. 2006. Concurrent probabilistic planning in the graphplan framework. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Mausam, and Kolobov, A. 2012. *Planning with Markov Decision Processes*. Morgan & Claypool.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. Lp-based heuristics for cost-optimal planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From non-negative to general operator cost partitioning. In *Proc. of National Conference on Artificial Intelligence (AAAI)*, 3335–3341.

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Revisiting goal probability analysis in probabilistic planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Teichteil-Königsbuch, F. 2012. Stochastic safest and shortest path problems. In *Proc. AAAI Conf. on Artificial Intelligence*.

Trevizan, F. W.; Thiébaux, S.; Santana, P. H.; and Williams, B. C. 2016. Heuristic search in dual space for constrained stochastic shortest path problems. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An lp-based heuristic for optimal planning. In *Int. Conf. on Principles and Practice of Constraint Programming*.