Operator Counting Heuristics for Probabilistic Planning

Felipe Trevizan, Sylvie Thiébaux, Patrik Haslum
Research School of Computer Science, Australian National University
Data61, CSIRO
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. In this paper, we present a generalization of the operator-counting family of heuristics to Stochastic Shortest Path problems (SSPs) that is able to represent the probability of the actions outcomes. Our experiments show that the equivalent of the *net change* heuristic in this generalized framework obtains significant run time and coverage improvements over other state-of-the-art heuristics in different planners.

1 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 *et al.*, 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). Many search algorithms have been developed for this purpose, including (L)TRDP [Barto *et al.*, 1995, Bonet and Geffner, 2003], LAO* [Hansen and Zilberstein, 2001], FRET [Kolobov *et al.*, 2011, Steinmetz *et al.*, 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 SSPs. 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.

In this paper, we present the *regrouped operator counting* heuristics (h^{roc}) [Trevizan *et al.*, 2017b] that, as far as we are aware, is the first domain-independent admissible heuristic for SSPs that reasons about both cost and outcomes probabilities of actions. h^{roc} is an extension of the operator-counting heuristics used in the deterministic setting [Pommerening *et al.*, 2014] in which additional constraints are added to model the outcome probability distribution of the each action. Our experiments show that iLAO* and LRTDP guided by h^{roc} often explore significantly fewer nodes than when guided by state-of-the-art heuristics for SSPs obtaining up to 56x speed up in running time. Moreover, h^{roc} is able to improve the scalability of the planners allowing them to solve larger problems than with the previous heuristics.

This paper focuses on one of the contributions in our ICAPS 2017 paper [Trevizan *et al.*, 2017b], and briefly summarises the others. We refer to that paper for further details.

2 Stochastic Shortest Path Problems

We start with some background about stochastic shortest paths problems, which we represent using a probabilistic variant of SAS$^+$ [Backström, 1992]. We then follow with a description of relevant solution methods and heuristics for SSPs.

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 $eff(a)$ of *effects*, each of which is a partial valuation over $\mathcal{V}$, and a *probability distribution* $\Pr_a(\cdot)$ over effects $e \in eff(a)$ 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, $\mathsf{G} \subset \mathsf{S}$ 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.

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 $\Pr(s'|s,a) = \sum_{e\in eff(a) \text{ s.t. } s'=res(s,e)} \Pr_a(e)$.

Policies. A solution for an SSP is a *policy* $\pi\colon \mathsf{S} \mapsto \mathsf{A}$ 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. In this paper, we assume that $s_0 \notin \mathsf{G}$ and that the goal is always reachable, i.e., that there are no dead ends. However, our experiments feature problems with dead ends and relax this assumption using the fixed-cost penalty formulation of dead ends [Kolobov *et al.*, 2012].[1]

The optimal policy π^* for an SSP might not be unique; however, any optimal policy can be obtained from the unique *optimal value function* V^* [Bertsekas and Tsitsiklis, 1991]. Given a state s, $V^*(s)$ represents the minimum total expected cost of reaching G from s and it is formally defined as the fixed-point solution of the Bellman equations:

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

for $s \in \mathsf{S} \setminus \mathsf{G}$ and $V^*(s) = 0$ for $s \in \mathsf{G}$.

Heuristic Search. Directly solving the Bellman equations (1) requires exploring the entire state space at once. In contrast, heuristic search algorithms for SSPs such as (i)LAO* [Hansen and Zilberstein, 2001] and LRTDP [Bonet and Geffner, 2003] 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).

All-outcomes determinisation. A key technique to compute heuristics for SSPs is the all-outcomes determinisation [Jimenez *et al.*, 2006]. Formally, given a probabilistic SAS$^+$ task, its all-outcomes determinisation is the deterministic SAS$^+$ task with identical set of variables, initial state, and goal, but whose actions are split into one deterministic action $\alpha_{a,e}$ for each probabilistic action $a \in \mathsf{A}$ and effect $e \in eff(a)$, such that $pre(\alpha_{a,e}) = pre(a)$, $eff(\alpha_{a,e}) = \{e\}$, and $C(\alpha_{a,e}) = C(a)$.

Current Heuristics for SSPs. The admissible heuristics (i.e.,lower bounds on V^*) used by heuristic search algorithms are typically obtained in two steps: (i) compute the all-outcomes determinisation of the given SSP; (ii), 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]. Unfortunately, these relaxations of V^* do not take probabilities into account, foregoing valuable information.

[1] More principled treatments of dead ends are also possible [Trevizan *et al.*, 2017a].

3 Regrouped Operator Counting Heuristics

In this section, we present the *Regrouped Operator-Counting Heuristic* h^{roc}, our probabilistic version of the family of operator-counting heuristics. This family of heuristics are described using a linear program (LP) of variables known as *operator counts* [Pommerening *et al.*, 2014]. When applied to the all-outcomes determinisation of a given probabilistic SAS$^+$ task, an operator count variable $Y_{a,e}$ represents, for each action a and effect e of a, the number of times a is executed and e occurs. These operator counts variables $Y_{a,e}$ are used in linear constraints to represent a relaxation of the original problem and an LP is formulated to find the solution with minimum cost for this relaxed problem. The idea behind h^{roc} is to add a set of linear constraints to any operator-counting heuristic that regroup the operator counts $Y_{a,e}$ of the same probabilistic action a and enforce the relationship between the respective probabilities of the effects e of a.

In this paper, we focus on the net change heuristic h^{net}, that is, the operator-counting heuristic using *net change constraints*. Intuitively, the net change heuristic keeps track of the changes in the value of each state variable from a state to another. 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 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 defined as the linear constraints (C1) and (C2) and the net change heuristic h^{net} is formally described in Definition 1.

$$\sum_{(a,e)\in AP_{v=d}} Y_{a,e} - \sum_{(a,e)\in AC_{v=d}} Y_{a,e} + \sum_{(a,e)\in SP_{v=d}} Y_{a,e} \geq \min pnc^{s\to s_\star}_{v=d} \qquad \text{(C1)}$$

$$\sum_{(a,e)\in AP_{v=d}} Y_{a,e} - \sum_{(a,e)\in AC_{v=d}} Y_{a,e} - \sum_{(a,e)\in SC_{v=d}} Y_{a,e} \leq \max pnc^{s\to s_\star}_{v=d} \qquad \text{(C2)}$$

Definition 1 (net change heuristic). *Given a probabilistic SAS$^+$ task, the net change heuristic h^{net} at state s is the solution of the LP:*

$$h^{net}(s) = \min_Y \sum_{a,e} Y_{a,e} C(a) \;\Big|\; \mathsf{N}^{v,d,s}\ \forall v \in \mathcal{V}, d \in \mathsf{D}_v$$

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 linear constraints:

Definition 2 (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 their dependency, i.e., $Y_{a,e_i} > 0$ iff $Y_{a,e_j} > 0$ for all $\{e_1, e_2\} \subseteq eff(a)$.

The heuristic h^{roc} is presented in Definition 3. h^{roc} is an admissible heuristic for SSPs, that is, for all $s \in \mathsf{S}$, $h^{\text{roc}}(s) \leq V^*(s)$ [Trevizan *et al.*, 2017b]. Intuitively, the admissibility of h^{roc} is due the admissibility of h^{net} and the fact the only difference between them is the set of regroup constraints that enforces the probabilistic definition of actions through the ratio of their expectations; therefore, as in the original SSP, if a particular effect of an action is desirable, all other effect of the same action must be accounted for since they will happen with positive probability.

Definition 3 (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:*

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

4 Empirical Evaluation

In this section we empirically evaluate h^{roc} against the following state-of-the-art heuristics for SSPs: $h^{\max}$, h^{lmc} and net change heuristic h^{net}. Notice that all these heuristics are determinisation-based heuristics. We use LRTDP and iLAO* as the search algorithms for this comparison. Each parametrization of planner and heuristic solves the same problem 30 times using a different random seed on each run to initialize the planner to account for the stochastic nature of the problem. For each run, we enforce a 30-minutes and 4-Gb cut-off for all experiments. We use two metrics for our experiments: (i) coverage, the number of runs a given parametrization found the optimal solution (out of 30) for each problem; and (ii) runtime, the average time spent to find the optimal solution *over the runs that found the optimal solution*.

We consider the following domains from the 2008 International Planning Competition (IPC'08): probabilistic Blocks World, Exploding Blocks World, and Triangle Tire World. We also consider a new domain, *Probabilistic Parc Printer*. This domain is a 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.

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

Table 1 presents coverage results for a subset of our experiments. The following is a summary of our findings from the experiments and we refer the reader to Trevizan *et al.* (2017b) for a comprehensive description of our methodology, domains and results.

Does taking probability into account in the heuristic help? Yes. Notice that the only difference between h^{net} and h^{roc} is that h^{roc} takes probability into account through the regrouping constraints and planners using h^{roc} obtained a speed up w.r.t. to h^{net} between 2x-56x, 1.3x-10x, and 1.1x-14x for blocks world, tire world and parc printer 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 statistical difference – unless we incorporate dead-end detection as reflected in the table and explained below.

How does h^{roc} compare to the state-of-the-art? For blocks world, planners using 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. 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}. Except in exploding blocks world, we found that h^{roc} expands 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} and h^{roc} 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} 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. The results for $h^{\text{roc}}_{\text{de}}$ corroborate the above explanation because of the large increase in performance when compared against h^{roc}. 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 the difference is statistically insignificant for 2 problems.

5 Beyond SSPs and h^{roc}

C-SSPs. Constrained SSPs (C-SSPs) are a natural generalization of SSPs to model planning under uncertainty problems for resource-bounded agents with multiple competing objectives [Altman, 1999, Dolgov and Durfee, 2005]. In a C-SSP, actions are associated with multiple cost functions (e.g., fuel, time, money), one of which is designated as the primary, and the others as secondary costs, and one seeks a policy that minimizes the expected primary cost subject to cost constraints, i.e., constraints over the expected secondary costs.

h^{roc} for C-SSPs. Recently, the first heuristic search algorithm for C-SSPs, i-dual, was introduced [Trevizan *et al.*, 2016]. Although it provided a large improvement over blind search algorithms, its full potential was not realised due to the lack of heuristics that could take cost constraints into account. We have shown how h^{roc} can be easily extended to incorporate such constraints resulting in $h^{\text{c-roc}}$ [Trevizan *et al.*, 2017b], the first heuristic for C-SSPs that reasons about both outcome probabilities of actions and cost constraints. The empirical evaluation of i-dual using h^{roc} against $h^{\text{c-roc}}$ shows that taking cost constraints into consideration improves both the scalability and running time of i-dual, e.g., $h^{\text{c-roc}}$ successfully solved 16 problems from the constrained version of the parc printer domain that h^{roc} could not solve.

Projection Occupation Measure Heuristic. Similarly to h^{roc}, the *projection occupation measure heuristic (h^{pom})* [Trevizan *et al.*, 2017b] is a heuristic for SSPs that takes outcome probabilities of actions into consideration. h^{pom} is also defined as an LP and is formulated using occupation measures, which are the probabilistic counterpart of operator counts. Formally, an occupation measure $x_{s,a}$ represents the expected number of times action a is executed in state s and the *dual LP* representation of the Bellman equations is formulated using them [D'Epenoux, 1963].

Given a probabilistic SAS$^+$ task, h^{pom} works by projecting the problem onto its variables and representing each projection as its own SSP using occupation measures. A benefit of projections is that they are still probabilistic problems; therefore the outcome probabilities of actions is not lost. Nonetheless, treating the projections as independent problems yields a lower bound on V^* worse than the state-of-the-art heuristics based on determinisation [Trevizan *et al.*, 2017b]. Instead, h^{pom} weakly ties all projections together to obtain a relaxed problem that can still be solved efficiently while providing a tighter lower bound on V^*. This weak tying is implemented as a set of linear constraints over the occupation measures enforcing that the total expected number of times a given action is executed in each projection is the same. We proved that h^{pom} is admissible and that, for all $s \in \mathsf{S}$, $h^{\text{roc}}(s) \leq h^{\text{pom}}(s)$. Our experiments show that iLAO* and LRTDP guided by h^{roc} are more efficient then their counterparts using h^{pom}. This stems from the fact that h^{roc} requires substantially fewer LP variables than h^{pom}. Similarly to h^{roc}, h^{pom} can be generalized to C-SSPs by adding cost constraints into its LP formulation.

Integrated i-dual. An advantage of occupation measure heuristics such as h^{pom} over operator counting ones such as h^{roc}, 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 current policy at each iteration. This yields a new algorithm, integrated i-dual (i^2-dual) [Trevizan *et al.*, 2017b], which represents 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. In our experiments, i^2-dual outperforms i-dual in coverage, time and number of nodes expanded, regardless of the heuristic used by the latter. For instance, in the constrained version of the parc printer domains, i^2-dual obtained a coverage between 30% and 100% in 13 problems for which all other planners' coverage was 0% and up to 34x speed up w.r.t. the second best planner in the other problems.

6 Conclusion

In this paper, we have presented h^{roc} and the first domain-independent admissible heuristic specifically designed to exploit the interactions between probabilities and action costs found in SSPs. We have shown that h^{roc} 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 h^{roc} by augmenting its 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].

Acknowledgements

This research was funded by AFOSR grant FA2386-15-1-4015.

References

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

[Backström, 1992] Christer Backström. Equivalence and tractability results for SAS$^+$ planning. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, 1992.

[Barto *et al.*, 1995] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artif. Intell.*, 72(1-2):81–138, 1995.

[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.

[Bonet and Geffner, 2001] Blai Bonet and Hector Geffner. Planning as heuristic search. *Artif. Intell.*, 129(1-2):5–33, 2001.

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

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

[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.

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

[Haslum and Geffner, 2000] Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In *Proc. Int. Conf. of Artificial Intelligence Planning Systems*, pages 140–149, 2000.

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

[Helmert *et al.*, 2007] Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*, pages 176–183, 2007.

[Jimenez *et al.*, 2006] Sergio Jimenez, Andrew Coles, and Amanda Smith. Planning in probabilistic domains using a deterministic numeric planner. In *Proc. Workshop of the UK Planning and Scheduling Special Interest Group*, 2006.

[Kolobov *et al.*, 2011] Andrey Kolobov, Mausam, Daniel S. Weld, and Hector Geffner. Heuristic search for generalized stochastic shortest path mdps. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2011.

[Kolobov *et al.*, 2012] Andrey Kolobov, Mausam, and Daniel S. Weld. A theory of goal-oriented mdps with dead ends. In *Proc. Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2012.

[Little and Thiébaux, 2006] Iain Little and Sylvie Thiébaux. Concurrent probabilistic planning in the graphplan framework. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2006.

[Pommerening *et al.*, 2014] Florian Pommerening, Gabriele Röger, Malte Helmert, and Blai Bonet. Lp-based heuristics for cost-optimal planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2014.

[Pommerening *et al.*, 2015] Florian Pommerening, Malte Helmert, Gabriele Röger, and Jendrik Seipp. From non-negative to general operator cost partitioning. In *Proc. of National Conference on Artificial Intelligence (AAAI)*, pages 3335–3341, 2015.

[Steinmetz *et al.*, 2016] Marcel Steinmetz, Joerg Hoffmann, and Olivier Buffet. Revisiting goal probability analysis in probabilistic planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2016.

[Trevizan *et al.*, 2016] Felipe W. Trevizan, Sylvie Thiébaux, Pedro Henrique Santana, and Brian C. Williams. Heuristic search in dual space for constrained stochastic shortest path problems. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2016.

[Trevizan *et al.*, 2017a] Felipe W. Trevizan, Florent Teichteil-Kœnigsbuch, and Sylvie Thiébaux. Efficient solutions for stochastic shortest path problems with dead ends. In *Proc. 33rd Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2017.

[Trevizan *et al.*, 2017b] Felipe W. Trevizan, Sylvie Thiébaux, and Patrik Haslum. Occupation measure heuristics for probabilistic planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2017.

[van den Briel *et al.*, 2007] Menkes van den Briel, J. Benton, Subbarao Kambhampati, and Thomas Vossen. An lp-based heuristic for optimal planning. In *Int. Conf. on Principles and Practice of Constraint Programming*, 2007.