Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Revisiting Goal Probability Analysis in Probabilistic Planning

Marcel Steinmetz and **Jörg Hoffmann**
Saarland University
Saarbrücken, Germany
{steinmetz,hoffmann}@cs.uni-saarland.de

Olivier Buffet
INRIA / Université de Lorraine
Nancy, France
olivier.buffet@loria.fr

Abstract

Maximizing goal probability is an important objective in probabilistic planning, yet algorithms for its optimal solution are severely underexplored. There is scant evidence of what the empirical state of the art actually *is*. Focusing on heuristic search, we close this gap with a comprehensive empirical analysis of known and adapted algorithms. We explore both, the general case where there may be 0-reward cycles, and the practically relevant special case of acyclic planning, like planning with a limited action-cost budget. We consider three different algorithmic objectives. We design suitable termination criteria, search algorithm variants, dead-end pruning methods using classical planning heuristics, and node selection strategies. Our evaluation on more than 1000 benchmark instances from the IPPC, resource-constrained planning, and simulated penetration testing reveals the behavior of heuristic search, and exhibits several improvements to the state of the art.

Introduction

Goal probability maximization in MDPs is important in planning scenarios ranging from critical decision-making (e. g. maximizing the probability to survive) to security tests (analyzing the chances that an attacker may compromise a valuable asset), and generally in problems with *unavoidable dead-ends* (e. g. (Kolobov et al. 2011; Kolobov, Mausam, and Weld 2012; Teichteil-Königsbuch 2012)). The objective partly underlies the International Probabilistic Planning Competition (IPPC) (Younes et al. 2005; Bryce and Buffet 2008; Coles et al. 2012), when planners are evaluated by how often they reach the goal in online policy execution.

We consider here the optimal offline setting, i. e., computing the exact maximum goal probability. We refer to this objective as *MaxProb*. While MaxProb certainly is relevant, there has been little work towards developing solvers. Optimal MDP heuristic search (Barto, Bradtke, and Singh 1995; Hansen and Zilberstein 2001; Bonet and Geffner 2003; McMahan, Likhachev, and Gordon 2005; Smith and Simmons 2006; Bonet and Geffner 2006) has been successful in expected-cost minimization, but suffers from a lack of admissible heuristic estimators of goal probability. The best known possibility is to detect *dead-ends* and set their estimate to 0, using the trivial estimate 1 elsewhere. Another major obstacle are complications arising from 0-reward cycles. As pointed out by Kolobov et al. (2011), MaxProb is equivalent to a non-discounted reward maximization problem, not fitting the stochastic shortest path (SSP) framework (Bertsekas 1995) because non-goal cycles receive 0 reward and thus improper policies do not accumulate reward $-\infty$.

Kolobov et al. propose *FRET (find, revise, eliminate traps)*, which admits heuristic search, but necessitates several iterations of complete searches, in between which FRET eliminates 0-reward cycles (traps). Hou et al. (2014) consider several variants of topological VI (Dai et al. 2011), solving MaxProb but necessitating to build the entire reachable state space. Kolobov et al. (2012) and Teichteil (2012) consider objectives asking for the cheapest policy among those maximizing goal probability, also requiring FRET or VI. Other works addressing goal probability maximization (e. g. (Teichteil-Königsbuch, Kuter, and Infantes 2010; Camacho et al. 2016)) do not aim at guaranteeing optimality. In summary, heuristic search for MaxProb is challenging, and has only been addressed by Kolobov et al. (2011).

Kolobov et al.'s experiments run only one configuration of search (LRTDP (Bonet and Geffner 2003)), with one possibility for dead-end detection (SixthSense (Kolobov, Mausam, and Weld 2010)), on a single domain (Exploding-Blocks). This outperforms VI, but the dead-end detection is not used in VI so it is unclear to what extent this is due to the actual heuristic search, rather than the state pruning itself.

Given this: *(i) What is actually the empirical state of the art in heuristic search for MaxProb? Are there other known algorithms, or variants thereof, that work better? (ii) What about simpler special cases, and weaker objectives, that are still practically relevant but that may be easier to solve?*

Question (ii) is interesting because such special cases and weaker objectives do indeed exist. A practically relevant special case is probabilistic planning with acyclic state spaces. This applies, e. g., to IPPC TriangleTireworld. More importantly, planning with a limited action-cost budget, *limited-budget planning*, is acyclic when action costs are non-0, strictly decreasing the remaining budget. Furthermore, simulated *penetration testing (pentesting)*, as per Hoffmann (2015), is acyclic. The MDP there models a network intrusion from the point of view of an attacker, which is acyclic because each exploit can be attempted at most once. In acyclic problems, there are no 0-reward cycles so we are facing an SSP problem and the need for FRET disappears.

Regarding weaker objectives, in addition to MaxProb, it is relevant to ask whether the maximum goal probability exceeds a given threshold θ, or to require computing the maximum goal probability up to a given accuracy δ. We refer to these objectives as *AtLeastProb* and *ApproxProb* respectively. For example, in pentesting AtLeastProb naturally assesses the level of network security: Can an attacker reach a target host (e. g. a customer data server) with probability greater than a given security margin (e. g. 0.01)?

Both AtLeastProb and ApproxProb allow early termination based on maintaining a lower (pessimistic) bound V^L in addition to the upper (admissible/optimistic) bound V^U. This is especially promising in AtLeastProb where we can terminate if the lower bound already is good enough ($V^L \geq \theta$) or if the upper bound already proves infeasibility ($V^U < \theta$). Good anytime behavior, on either or both bounds, translates into early termination.

To answer our research questions (i) and (ii), we design an algorithm space characterized by (a) *search algorithm*, (b) *dead-end pruning method*, and (c) *node selection strategy*. For (a), we design variants of AO* and LRTDP maintaining upper and lower bounds for early termination, and we design a variant of FRET better suited to problems with uninformative initial upper bounds. We furthermore design a new probabilistic-state-space reduction method, via determinized bisimulation. For (b), we employ classical-planning heuristic functions, a connection not made before and which is especially promising in limited-budget planning where we can prune against the remaining budget.[1] For (c), we design a comprehensive arsenal of simple strategies biasing node selection to foster early termination.

Our techniques are implemented in FD (Helmert 2006). We explore their behavior on a benchmark suite including domains from the IPPC, resource-constrained planning, and pentesting, comprising 1089 instances in total. Amongst other things, we observe: substantial benefits of heuristic search, even with trivial initial estimates (+9% total coverage), more so with initial estimates based on dead-end detection (+12%); substantial benefits of early termination (e. g. for AtleastProb +8% with $\theta = 0.2$ and +7% with $\theta = 0.9$); and dramatic benefits of our FRET variant (+32%). Our state-space reduction method yields an optimal MaxProb solver that scales just as well in TriangleTireworld as the sub-optimal solver Prob-PRP (Muise, McIlraith, and Beck 2012; Camacho et al. 2016) – yet not only for the standard version where the goal can be achieved with certainty, but also for the limited-budget version where that is not so.

We omit some technical details. Full details are available in an online TR (Steinmetz, Hoffmann, and Buffet 2016).

MDP Models

We consider a probabilistic extension of STRIPS, in two variants, with respectively without a limited action-cost budget. We specify first the unlimited-budget version. Planning *tasks* are tuples $\Pi = (F, A, I, G)$ consisting of a finite set F of *facts*, a finite set A of *actions*, an *initial state* $I \subseteq F$, and a *goal* $G \subseteq F$. Each $a \in A$ is a pair $(pre(a), O(a))$ where $pre(a) \subseteq F$ is the *precondition*, and $O(a)$ is the finite set of *outcomes* o, each being a tuple $(p(o), add(o), del(o))$ of *outcome probability* $p(o)$, *add list* $add(o) \subseteq F$, and *delete list* $del(o) \subseteq F$. We require that $\sum_{o \in O(a)} p(o) = 1$.

The *state space* of a task Π is a probabilistic transition system $(S, P, I, S_\top)$. Here, S is the set of *states*, each $s \in S$ associated with its set $F(s)$ of true facts. The initial state I is that of Π, the set of *goal states* $S_\top \subseteq S$ contains those s where $G \subseteq F(s)$. The *transition probability* function $P : S \times A \times S \mapsto [0, 1]$ is defined as follows. Action a is *applicable* to state s if $s \notin S_\top$ (goal states are absorbing) and $pre(a) \subseteq F(s)$. By $s[\![o]\!]$ we denote the result of outcome o in s, i. e., $F(s[\![o]\!]) = (F(s) \cup add(o)) \setminus del(o)$. $P(s, a, t)$ is $p(o)$ if a is applicable to s and $t = s[\![o]\!]$,[2] and is 0 otherwise (there is no transition). *Absorbing states* are those with no outgoing transitions (no applicable actions). The set of non-goal absorbing states – *lost states* – is denoted $S_\bot$.

For limited-budget planning, we extend the above as follows. A *limited-budget task* is a tuple $\Pi = (F, A, I, G, b)$, as above but now with a *budget* $b \in \mathbb{R}^+_0$, and each outcome o being associated with a *cost* $c(o) \in \mathbb{R}^+_0$. In addition to their true facts $F(s)$, states s are also associated with their *remaining budget* $b(s) \in \mathbb{R}$. States with negative remaining budget $b(s) < 0$ are legal and may occur, but are lost, $s \in S_\bot$, because: the goal states $s \in S_\top$ are those where $G \subseteq F(s)$ *and* $b(s) \geq 0$; the actions a applicable to s are those where $pre(a) \subseteq F(s)$ *and* at least one outcome fits within the remaining budget, i. e., there exists $o \in O(a)$ so that $c(o) \leq b(s)$. In the outcome states $s[\![o]\!]$, the outcome's cost is deduced from the budget, i. e., $b(s[\![o]\!]) = b(s) - c(o)$.

Note here that, if $c(o) > 0$ for all o, then the state space is acyclic because every transition strictly reduces the remaining budget. Note further that the remaining budget is local to each state. If some states in a policy violate the budget, other parts of the policy (even other outcomes of the same action) can still continue trying to reach the goal. This differs from constrained MDPs (Altman 1999), where the budget bound is applied globally to the expected cost of the policy.

Compared to planning with continuous resource-consumption uncertainty (Marecki and Tambe 2008; Meuleau et al. 2009; Coles 2012), our form of limited-budget probabilistic planning is restrictive. Yet it is still natural and relevant, suiting for example any problem asking to achieve a goal within a given number of steps.

A *policy* is a partial function $\pi : S \setminus (S_\top \cup S_\bot) \mapsto A \cup \{*\}$, mapping each non-absorbing state s within its domain either to an action applicable in s, or to the *don't care* symbol $*$. The latter will be used (only) by policies that already achieve sufficient goal probability elsewhere, so do not need to elaborate on how to act on s and its descendants. That is, we still require *closed* policies, and use $*$ to explicitly indicate special cases where actions may be chosen arbitrarily. For-

[1] On the side, we discover that Domshlak and Mirkis' (2015) landmarks compilation is, per se, equivalent to such pruning.

[2] We assume here that each $o \in O(a)$ leads to a different outcome state. This is just to simplify notation (our implementation does not make this assumption).

mally, $\pi(s) = *$ extends the domain of π by picking, for every $t \notin S_\top \cup S_\perp$ reachable from s and where $\pi(t)$ is undefined, an arbitrary action a applicable in t and setting $\pi(t) := a$. A policy π is *closed for state* s if, for every state $t \notin S_\top \cup S_\perp$ reachable from s under π, $\pi(t)$ is defined; π is *closed* if it is closed for the initial state I.

Following Kolobov et al. (2011), we formulate goal probability as maximal non-discounted expected reward where reaching the goal gives reward 1 and all other rewards are 0. The value $V^\pi(s)$ of a policy π closed for state s then is:

$$V^\pi(s) = \begin{cases} 1 & s \in S_\top \\ 0 & s \in S_\perp \\ \sum_t P(s, \pi(s), t) V^\pi(t) & \text{otherwise} \end{cases}$$

The value of state s is $V^*(s) = \max_{\pi : \pi \text{ closed for } s} V^\pi(s)$.

For acyclic state spaces, we are facing an SSP problem (every run ends in an absorbing state in a finite number of steps). For cyclic state spaces, the Bellman update operator may have multiple sub-optimal fixed points, and updates from an optimistic (upper-bound) initialization are not guaranteed to converge to the optimum. One can either use a pessimistic initialization, or Kolobov et al.'s FRET method.

We consider three different *objectives* (algorithmic problems) for goal probability analysis:

MaxProb: Find an optimal policy, i. e., a closed π s.t. $V^\pi(I) = V^*(I)$.

AtLeastProb: Find a policy guaranteeing a user-defined goal probability threshold $\theta \in [0, 1]$, i. e., a closed π s.t. $V^\pi(I) \geq \theta$. (Or prove that such π does not exist.)

ApproxProb: Find a policy optimal up to a user-defined goal probability accuracy $\delta \in [0, 1]$, i. e., a closed π s.t. $V^*(I) - V^\pi(I) \leq \delta$.

We next examine search algorithms, pruning methods, and node selection strategies, to solve these problems.

Search Algorithms

We use VI as a baseline, and design variants of AO* and LRTDP. For VI, we make one forward pass building the reachable state space (actually its pruned subset, see next section). We initialize the value function as 0 everywhere. For acyclic cases, we then perform a single backward pass of Bellman updates, starting at absorbing states and updating children before parents. For the general case, we assume a parameter ϵ and run topological VI (Dai et al. 2011): We find the strongly connected components (SCC) of the state space, and handle each SCC individually, children SCCs before parent SCCs. VI on an SCC stops when every state is *ϵ-consistent*, i. e. when its Bellman residual is at most ϵ.[3]

For AO*, we restrict ourselves to the acyclic case, where the overhead for repeated value iteration fixed points, inherent in LAO* (Hansen and Zilberstein 2001), disappears. Figure 1 shows pseudo-code. The algorithm incrementally constructs a subgraph Θ of the state space. The handling

procedure *GoalProb-AO**
initialize Θ to consist only of I; *Initialize*(I)
loop do
  if [MaxProb: $V^L(I) = 1$]
    [AtLeastProb: $V^L(I) \geq \theta$]
    [ApproxProb: $V^L(I) \geq 1 - \delta$ or $V^U(I) - V^L(I) \leq \delta$] **then**
    return π^L **endif** /* early termination (positive) */
  if [AtLeastProb: $V^U(I) < \theta$] **then**
    return "impossible" **endif** /* early termination (negative) */
  if ex. leaf state $s \notin S_\top \cup S_\perp$ in Θ reachable using π^U **then**
    select such a state s
  else return π^U **endif** /* regular termination */
  for all a and t where $P(s, a, t) > 0$ **do**
    if t not already contained in Θ **then**
      insert t as child of s into Θ; *Initialize*(t)
    else insert s as a new parent of t into Θ **endif**
  endfor
  BackwardsUpdate(s)
endloop
procedure *Initialize*(s):
$V^U(s) := \begin{cases} 0 & s \in S_\perp \\ 1 & \text{otherwise} \end{cases}$
$V^L(s) := \begin{cases} 1 & s \in S_\top \\ 0 & \text{otherwise} \end{cases}$
if $s \notin S_\top \cup S_\perp$ **then** $\pi^L(s) := *$ **endif**

Figure 1: AO* for MaxProb, AtLeastProb, and ApproxProb (as indicated), on acyclic state spaces. π^U is the current greedy policy on V^U, π^L is the current greedy policy on V^L. *BackwardsUpdate*(s) updates all of V^U, π^U, V^L, π^L. As states may have several parents in Θ, we first make a backwards sweep to collect the sub-graph $\Theta|_s$ ending in s. Then we update $\Theta|_s$ in reverse topological order.

of duplicates is simple, identifying search nodes with states, as the state space is acyclic. For the same reason, simple backward updating suffices to maintain the value function.

Adopting ideas from prior work (e. g. (McMahan, Likhachev, and Gordon 2005; Little, Aberdeen, and Thiébaux 2005; Smith and Simmons 2006)), we maintain two value functions, namely both an upper bound V^U and a lower bound V^L on goal probability. Both are initialized trivially, for lack of heuristic estimators of goal probability (dead-end detection, as a simple but non-trivial V^U initialization, will be discussed in the next section). Nevertheless, both bounds can be useful for search. To refute an action, it often suffices to reduce V^U for just one of its outcomes. Hence, even for trivial initialization, V^U may allow to disregard parts of the search space, in the usual way of admissible heuristic functions. As we shall see, this kind of behavior occurs frequently. Furthermore, there are various possibilities for early termination. The lower bound enables positive early termination when we can already guarantee sufficient goal probability, namely 1 (MaxProb), θ (AtLeastProb), or $1 - \delta$ (ApproxProb). The upper bound enables negative early termination in AtLeastProb, when $V^U < \theta$. In ApproxProb, clearly we can terminate when $V^U(I) - V^L(I) \leq \delta$.[4]

[3] *Focused* topological VI eliminates sub-optimal actions in a pre-process to obtain smaller SCCs. While this can be much more runtime-effective, it still requires to build the entire state space, doing which was the *only* reason for VI failures in our experiments. So we do not consider focused topological VI here.

[4] The $V^L = 1$ and $V^L(I) \geq 1 - \delta$ criteria are redundant when maintaining an upper bound, i. e., for heuristic search, where they are subsumed by regular termination respectively termination on $V^U(I) - V^L(I) \leq \delta$. In configurations not maintaining V^U, how-

Regarding correctness: Trivially, $V^U(s)$ and $V^L(s)$ indeed are upper respectively lower bounds on the goal probability of the states s in Θ, at any point in time. Furthermore, π^L is always a closed policy, because it applies the don't care symbol $*$ at the non-absorbing leaf states in Θ (note also that $*$ is applied *only* on those states). Its goal probability $V^{\pi^L}(s)$ is at least the lower-bound goal probability, $V^{\pi^L}(s) \geq V^L(s)$, because $V^L(s)$ is monotonic.

For LRTDP, we consider the general case, including cyclic state spaces. We omit the pseudo-code, for space reasons (it is available in the TR). Like in GoalProb-AO*, we maintain V^L in addition to V^U. We test the exact same early termination criteria. Note that this is valid even in the general/cyclic case, i. e., if early termination applies then we can terminate the overall FRET process. We include an additional stopping criterion for trials in the cyclic case, also used by Kolobov et al. (2011), stopping if the current state s is ϵ-consistent. This keeps trials from getting trapped in 0-reward cycles, yet preserves the property that, upon regular termination, all states reachable using π^U are ϵ-consistent.

In the cyclic case, the V^U fixed point found by LRTDP may be sub-optimal, so we have to use FRET. In the acyclic case, we use $\epsilon = 0$, and a single call to LRTDP suffices.

To study early termination capabilities, for X $\in$ {AO*, LRTDP} we will consider variants $X|_U$ and $X|_L$, maintaining only V^U respectively only V^L. Early termination criteria involving the non-maintained bound are disabled. We write $X|_{LU}$ to make explicit that both bounds are used. For AO$^*|_L$, all non-absorbing leaf states in Θ are open (rather than only those reachable using π^U), and in case of regular termination we return π^L. We do not consider a variant LRTDP$|_L$ as LRTDP without an upper bound does not make sense.

We finally design a variant of FRET, and a new state-space reduction method. FRET performs an iteration of heuristic searches, each finding a fixed point of V^U. In between iterations, FRET runs a *trap elimination* step, which finds non-goal cycles in the greedy-policy graph with respect to V^U, and forces the next search iteration to avoid these cycles. The "greedy-policy graph" here considers *all* actions greedy with respect to V^U. We refer to this design as *FRET-*V^U. Our alternative design, *FRET-*π^U, instead considers the graph induced by the actions π^U selected into the current greedy policy. This provides the same convergence guarantee (see the TR for the proof). FRET-V^U may require less iterations, yet each trap elimination step may be much more costly. In particular, in goal probability analysis, V^U often is 1 almost everywhere in the first step, and the graph considered by FRET-V^U is almost the entire reachable state space.

Our reduction method computes a bisimulation of the *all-outcomes determinization* (e. g. (Yoon, Fern, and Givan 2007; Little and Thiebaux 2007)), using standard *merge-and-shrink* methods (Helmert et al. 2014). We then run any MDP algorithm on the bisimulated state space. This is sound because bisimilar states have equivalent transition behavior, and transitions in the all-outcomes determinization are action outcomes in the original task. Thus bisimilar states are equivalent in the probabilistic state space (the bisimulation is a "homogenous partition" as per Dean and Givan (1997)).

ever, these termination criteria can be very useful to reduce search.

Dead-End Pruning

Dead-ends are states s where $V^*(s) = 0$. One can treat such s exactly like lost states $S_\perp$ (except for setting $\pi^L(s) := *$). Apart from this pruning itself, for the heuristic search algorithms this provides a non-trivial initialization of V^U, typically leading to additional search reductions.

Kolobov et al. (2011) employ SixthSense (Kolobov, Mausam, and Weld 2010), which learns dead-end detection rules by generalizing from information obtained using a classical planner. Here we instead exploit the power of classical-planning heuristic functions run on the all-outcomes determinization, readily available in our FD implementation framework. This works especially well in limited-budget planning, where we can use lower bounds on determinized remaining cost to detect states with insufficient remaining budget. Note that this is natural and effective using admissible remaining-cost estimators, yet would be impractical using an actual planner (which would need to be optimal and thus prohibitively slow). For the general case, we can use any heuristic function able to detect dead-ends (returning ∞), which applies to most known heuristics.

We experiment with state-of-the-art heuristic functions, namely (a) an *admissible landmark heuristic* as per Karpas and Domshlak (2009), (b) *LM-cut* (Helmert and Domshlak 2009), (c) several variants of merge-and-shrink heuristics, and (d) $h^{\max}$ (Bonet and Geffner 2001) as a simple and canonical option. (a) turned out to perform consistently worse than (b), so we will report only on (b) – (d).

For limited-budget planning, we also considered to adopt the problem reformulation by Domshlak and Mirkis (2015) for oversubscription planning, which reduces the budget b using landmarks and in exchange allows to traverse yet non-used landmarks at a reduced cost during search. Somewhat surprisingly, however, pruning states whose reduced budget is < 0 is equivalent to the much simpler method pruning states whose heuristic (a) exceeds the remaining budget. The added value of Domshlak and Mirkis' reformulation thus lies, not in its pruning per se, but in its compilation into a planning language and the resulting combinability with other heuristics. We give more details in the TR.

Node Selection Strategies

In both GoalProb-AO* and GoalProb-LRTDP, good anytime behavior on V^L and/or V^U may translate into early termination. We explore the potential of fostering this via (1) biasing the tie-breaking in the selection of "best" actions π^U greedy with respect to V^U, and (2) biasing the choice of outcome states (AO*), respectively the outcome-state sampling during trials (LRTDP). AO$^*|_L$ is a special case where (1) does not apply but (2) is especially important as we are free to choose any open leaf state in the current search graph Θ.

We experimented with a variety of strategies. We give a brief summary only; details are available in the TR.

Our *default* strategy uses the standard in AO* and LRTDP. Tie-breaking for (1) is arbitrary, but fixed, i. e., $\pi^U(s)$ changes only if some other action becomes strictly better in

s. Open outcome states (2) in AO* are selected arbitrarily, and the bias in LRTDP is by outcome probability. Our *most-prob-outcome bias* strategy in AO* prefers likely outcomes.

Our *h-bias* strategy prefers states with smaller h value, where the heuristic h is the same one used for dead-end pruning. In (1), we break ties in favor of actions minimizing the expected heuristic value, in (2) we weigh (and renormalize) the outcome probabilities by $\frac{1}{h(t)} * P(s, a, t)$. Inspired by BRTDP (McMahan, Likhachev, and Gordon 2005), our *gap-bias* strategy breaks ties in (1) by maximal expected gap, and weighs the outcome probabilities in (2) by $[V^U(t) - V^L(t)] * P(s, a, t)$. Inspired by common methods in classical planning, e. g. (Hoffmann and Nebel 2001; Helmert 2006), our *preferred actions* strategy prefers in (1) actions used by delete-relaxed determinized plans.

In AO$^*|_{\mathrm{L}}$, the default strategy is depth-first, the rationale being to try to reach absorbing states quickly. The h-bias strategy selects a deepest leaf with minimal h value, the preferred actions strategy selects a deepest open leaf reachable using only preferred actions. We furthermore experiment with a breadth-first strategy, just for comparison.

Experiments

We implemented the algorithms in Fast Downward (FD) (Helmert 2006). FD's pre-processes were extended to handle PPDDL. The state of the art in optimal goal probability analysis is represented by particular points in our configuration space: topological VI and (FRET-V^U using) LRTDP$|_{\mathrm{U}}$ with dead-end pruning. The implementation by Kolobov et al. (2011), which uses the different dead-end detector SixthSense, is not available anymore (personal communication with Andrey Kolobov). The TR includes a detailed comparison against the results reported by Kolobov et al.

Our aim being to comprehensively explore the relevant problem space, we designed a broad suite of benchmarks, 1089 instances in total, based on domains from the IPPC, resource-constrained planning, and pentesting. From the IPPC, we selected those PDDL domains in STRIPS format, or with moderate non-STRIPS constructs easily compilable into STRIPS. This resulted in 10 domains from IPPC'04 – IPPC'08; we selected the most recent benchmark suite for each of these. For resource-constrained planning, we adopted the NoMystery, Rovers, and TPP benchmarks by Nakhost et al. (2012), more precisely those suites with a single consumed resource (fuel, energy, money), which correspond to limited-budget planning.[5] We created probabilistic versions by adding uncertainty about the underlying road map, akin to the Canadian Traveler scenario, each road segment being present with a given probability (this is encoded through a separate, probabilistic, action attempting a segment for the first time). For simplicity, we set that probability to 0.8 throughout. For pentesting, we modified the POMDP generator by Sarraute et al. (2012), which itself is based on a test scenario used at Core Security (http://www.coresecurity.com/). The generator uses a network consisting of an exposed part, a sensitive part, and a user part. It scales the numbers H of hosts and E of exploits. We modified the generator to output Hoffmann's (2015) *attack-asset MDP* pentesting models. Sarraute et al.'s POMDP model and solver (SARSOP (Kurniawati, Hsu, and Lee 2008), which is not optimal) scale to $H = 6, E = 10$. For our benchmarks, we fixed $H = E$ for simplicity (and to obtain a number of instances similar to the other benchmark domains). We scaled the instances from $6 \dots 20$ without budget limit, and from $10 \dots 24$ with budget limit.

From each of the above benchmark task Π, except the pentesting ones, we obtained several limited-budget benchmarks, as follows. We set outcome costs to 1 where not otherwise specified. We determined the minimum budget, $b_{\min}$, required to achieve non-0 goal probability. For the resource-constrained benchmarks, $b_{\min}$ is determined by the generator itself, as the minimum amount of resource required to reach the goal in the deterministic domain version. For all other benchmarks, we ran FD with A* and LM-cut on the all-outcomes determinization of Π. If this failed, we skipped Π, otherwise we read $b_{\min}$ off the cost of the optimal plan and created several limited-budget tasks $\Pi[C]$, differing in their *constrainedness level* C. Namely, following Nakhost et al. (2012), we set the global budget b in $\Pi[C]$ to $b := C * b_{\min}$, so that C is the factor by which the available budget exceeds the minimum needed (to be able to reach the goal at all). We let C range in $\{1.0, 1.2, \dots, 2.0\}$.

For AtleastProb, we let θ range in $\{0.1, 0.2, \dots, 1.0\}$ ($\theta = 0$ is pointless). For ApproxProb, we let δ range in $\{0.0, 0.1, \dots, 0.9\}$ ($\delta = 1$ is pointless). On cyclic problems, the convergence parameter ϵ was set to 0.00005 (the same value as used by Kolobov et al. (2011)). All experiments were run on a cluster of Intel E5-2660 machines running at 2.20 GHz, with time/memory cut-offs of 30 minutes/4 GB.

Acyclic Planning

We consider first acyclic planning. This pertains to all budget-limited benchmarks, to pentesting with and without budget limit, as well as to IPPC TriangleTireworld (moves can be made in only one direction so the state space is acyclic). We consider the 3 objectives MaxProb, AtLeastProb, and ApproxProb. We run all 6 search algorithm variants, each with up to 5 node selection strategies as explained. For dead-end pruning, we run LM-cut, as well as merge-and-shrink *(M&S)* with the state-of-the-art shrinking strategies based on bisimulation and an abstraction-size bound N; we show data for $N = \infty$ and $N = 100k$ (we also tried $N \in \{10k, 50k, 200k\}$ which resulted in similar behavior). We also run variants without dead-end pruning. We use the deterministic-bisimulation (BS) reduced state space with VI (the cases where BS succeeds are easily solved by VI). Overall, we get 217 different algorithm configurations.

We first examine the behavior of search algorithms and pruning on MaxProb. We fix the node selection to default, and we omit AO$^*|_{\mathrm{LU}}$ and LRTDP$|_{\mathrm{LU}}$ as, for MaxProb heuristic search, maintaining V^L is redundant (early termination is dominated by regular termination). Table 1 shows coverage

[5] To make the benchmarks feasible for optimal probabilistic planning, we had to reduce their size parameters (number of locations etc). We scaled all parameters with the same number < 1, chosen to get instances at the borderline of feasibility for VI.

Domain	#	VI				$AO^*\|_L$				$AO^*\|_U$				$LRTDP\|_U$				VI on BS
		–	LM	M&S N	M&S ∞	–	LM	M&S N	M&S ∞	–	LM	M&S N	M&S ∞	–	LM	M&S N	M&S ∞	
IPPC Benchmarks																		
TriaTire	10	4	4	4	4	4	4	4	4	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
IPPC Benchmarks with Budget Limit																		
Blocksw-b	66	24	**28**	24	24	24	**28**	24	24	24	**28**	24	24	24	**28**	24	24	24
Boxworl-b	18	0	**3**	0	0	0	**3**	0	0	0	**3**	0	0	0	**3**	0	0	0
Drive-b	90	**90**	**90**	**90**	52	**90**	**90**	**90**	52	**90**	**90**	**90**	52	**90**	**90**	**90**	52	52
Elevator-b	90	71	82	72	33	70	82	72	33	65	77	67	33	79	**86**	79	33	33
ExpBloc-b	84	32	46	38	37	32	46	38	37	39	57	39	37	38	**65**	39	37	37
Random-b	60	27	33	35	33	27	33	35	33	35	**44**	36	33	36	**44**	36	33	33
RecTire-b	36	30	31	**36**	**36**	30	31	**36**	**36**	30	31	**36**	**36**	30	31	**36**	**36**	**36**
Tirewor-b	90	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**	**90**
TriaTire-b	60	45	52	52	52	45	52	52	52	46	55	55	55	47	57	57	57	**60**
Zenotra-b	36	15	16	16	**18**	15	16	16	**18**	14	16	16	17	15	17	16	17	17
Probabilistic Resource-Constrained Benchmarks with Budget Limit																		
NoMystery-b	60	11	37	43	44	11	36	42	43	12	39	47	47	12	41	50	50	**51**
Rovers-b	60	23	39	31	40	23	38	31	40	23	44	33	45	25	46	35	46	**50**
TPP-b	60	18	35	25	25	16	35	24	24	15	37	26	22	19	**38**	27	25	26
Pentesting Benchmarks																		
Pentest	15	**9**	**9**	**9**	8	**9**	**9**	**9**	8	**9**	**9**	**9**	8	**9**	**9**	**9**	8	8
Pentest-b	90	57	**63**	62	37	57	**63**	62	37	57	**63**	**63**	37	57	**63**	**63**	37	37
$\sum$	925	546	658	627	533	543	656	625	531	559	693	641	546	581	**718**	661	555	564

Table 1: MaxProb coverage (number of tasks solved within time & memory limits) in acyclic planning. Best values in **boldface**. Domains "-b" modified with budget limit. "#": number of instances. "–": no pruning; else pruning, against remaining budget on "-b" domains, based on $h = \infty$ on other domains. "LM": LM-cut; "M&S": merge-and-shrink, "N" size bound $N = 100k$, "∞" no size bound. "VI on BS": VI run on reduced (bisimulated) state space.

data (runtime results are qualitatively similar, see the TR).

Of the pruning methods, LM-cut clearly stands out. For every search algorithm, it yields the by far best overall coverage. M&S has substantial advantages only in RectangleTireworld and NoMystery-b. Note that, for $N = \infty$, overall coverage is worse than for using no pruning at all. This is due to the prohibitive overhead, in some domains, of computing a bisimulation on the determinized state space. And, having invested this effort, it pays off more to use the bisimulation as a reduced MDP state space ("VI on BS"), rather than only for dead-end pruning. An extreme example is TriangleTireworld. Far beyond the standard benchmarks in Table 1 (triangle-side length 20), VI on BS scales to side length 74 in both the original domain and the limited-budget version. For comparison, the hitherto best solver by far was Prob-PRP (Camacho et al. 2016), which scales to side length 70 on the original domain,[6] and is optimal only for goal probability 1, i. e., in the presence of strong cyclic plans.

Of the search algorithms, $AO^*|_L$ is better than VI only in case of early termination on $V^L = 1$, when a full-certainty policy is found before visiting the entire state space. This happens very rarely here, and $AO^*|_L$ is dominated by VI (this changes for AtLeastProb, Figures 3 (a) and 4 below). $LRTDP|_U$ clearly outperforms $AO^*|_U$, presumably because it tends to find absorbing states more quickly.

To gauge the efficiency of heuristic search vs. blind search on MaxProb, compare $LRTDP|_U$ vs. VI in Table 1. Contrary to the intuition that a good initial goal probability estimator is required for heuristic search to be useful, $LRTDP|_U$ is clearly superior. Its advantage does grow with the quality of the initialization; LM-cut yields the largest coverage increase by far. However, even without dead-end pruning, i. e., with the trivial initialization of V^U, $LRTDP|_U$ dominates VI throughout, and improves coverage in 8 of the 16 domains. Figure 2 sheds additional light on this by comparing the respective search space sizes directly. The non-trivial initialization using LM-cut clearly helps. But even without it, gains of around 1 order of magnitude occur frequently, and larger gains (up to 3 orders of magnitude) occur in rare cases. As previously hinted, these observations have not been made in this clarity before: While Kolobov et al. (2011) also report LRTDP to beat VI on MaxProb, they consider only a single domain; they do not experiment with trivially initialized V^U; and they do not use dead-end pruning in VI, so that LRTDP already benefits from a smaller state space, and the impact of heuristic search remains unclear.

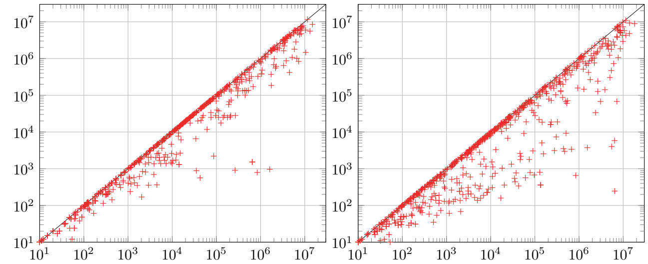

Figure 2: Number of states visited, for VI (x) vs. $LRTDP|_U$ (y), with no pruning (left) and with LM-cut pruning (right).

We now turn to the weaker objectives, AtLeastProb and ApproxProb. We fix LM-cut for the (almost always most effective) dead-end pruning. We examine the power of early termination for different search algorithms and node selection strategies. This is best viewed as a function of the goal probability threshold θ in AtLeastProb, and of the desired goal probability accuracy δ in ApproxProb. VI forms a baseline independent of θ. Consider Figure 3.

For AtLeastProb (Figure 3 (a)), one clear feature is again the superiority of LRTDP over AO^*. There is now the striking exception of $AO^*|_L$, however, which for small values of θ approaches (and in one case, surpasses) the performance of LRTDP. The depth-first expansion strategy is quite effective for anytime behavior on V^L and thus for termination via $V^L(I) \geq \theta$. It is way more effective than the heuristic search in $AO^*|_{LU}$. As we shall see below (Figure 4), it is often also more effective than LRTDP. In general, for all algorithms, using V^L is a clear advantage for small θ. For larger θ, maintaining V^L can become a burden, yet V^U is of advantage due to early termination on $V^U(I) < \theta$. Algorithms using both bounds exhibit an easy-hard-easy pattern.

The spike at the left-hand side in Figure 3 (a), i. e., significantly worse performance for $\theta = 0.1$ than for $\theta = 0.2$, is an outlier due to the Pentest domains (without them, $AO^*|_{LU}$ and $LRTDP|_{LU}$ exhibit a strict easy-hard-easy pattern). In contrast to typical probabilistic planning scenarios, in penetration testing the goal probability – the chances of a successful attack – are typically small, and indeed this is so in our benchmarks. Searches using an upper bound quickly ob-

[6] We could not run the limited-budget domain as Prob-PRP does not natively support a budget, and hard-coding the budget into PPDDL resulted in encodings too large to pre-process.

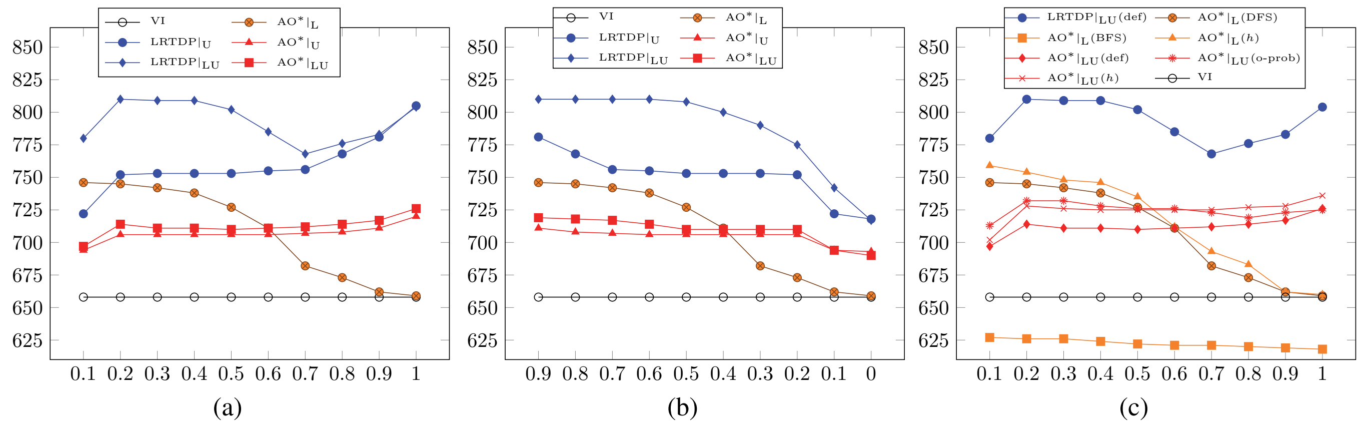

Figure 3: Total coverage in acyclic planning, for AtLeastProb as a function of θ in (a) and (c), for ApproxProb as a function of δ in (b). Node selection varies in (c), default used in (a) and (b). All configurations use LM-cut dead-end pruning.

tain $V^U(I) < 0.2$, terminating early based on $V^U(I) < \theta$ for $\theta = 0.2$. But it takes a long time to obtain $V^U(I) < 0.1$.

For ApproxProb (Figure 3 (b)), smaller values of δ consistently result in worse performance. We see again the superiority of LRTDP over AO*, with a similar though not as pronounced exception for AO*$|_\mathrm{L}$ in δ regions allowing aggressive early termination. We also see again the superiority of algorithms using both bounds over those that don't.

Figure 3 (c) examines the effect of different node selection strategies in AtLeastProb (the relative performance of node selection strategies is the same in ApproxProb, so we do not include a separate figure for that). For readability, we show only the most competitive base algorithms, AO*$|_\mathrm{L}$, AO*$|_\mathrm{LU}$, and LRTDP$|_\mathrm{LU}$ (as well as the VI baseline). For LRTDP, we show only default node selection, which consistently works a little better than the alternatives. For AO*$|_\mathrm{L}$, we see that the depth-first strategy is important (and way beyond breadth-first, which does worse than VI). The h-bias strategy is generally on par with depth-first. For AO*$|_\mathrm{LU}$, the main observation is that the most-prob-outcome bias is helpful, improving over the default strategy except for high values of θ. The h-bias consistently improves a bit on default AO*. The gap-bias and preferred actions strategies are not shown as they were consistently slightly worse (apparently, the gap-bias leads to a more breadth-first style behavior, while preferred actions mainly cause runtime overhead).

To conclude our discussion of acyclic planning, Figure 4 exemplifies typical anytime behavior, i. e., the development of the $V^L(I)$ and $V^U(I)$ bounds on the initial state value, as a function of runtime, for LRTDP$|_\mathrm{LU}$ and AO*$|_\mathrm{L}$ (using default node selection because the alternatives are not beneficial for these algorithms). The benefit of LM-cut pruning is evident. Observe that AO*$|_\mathrm{L}$ is way more effective than LRTDP in quickly improving the lower bound. Indeed, the runs shown here find an optimal policy very quickly. Across the benchmarks solved by both AO*$|_\mathrm{L}$ and LRTDP, omitting those where both took < 1 second, in 56% of cases AO*$|_\mathrm{L}$ finds an optimal policy faster than LRTDP. On (geometric) average, AO*$|_\mathrm{L}$ takes 66% of the time taken by LRTDP for this purpose. On the downside, unless $V^*(I) \geq \theta$, AO*$|_\mathrm{L}$ must explore the entire state space. Its runs in Figure 4 exhaust memory for MaxProb. In summary, heuristic search is much stronger in proving that the maximum goal probability is found, but is often distracting for improving V^L quickly.

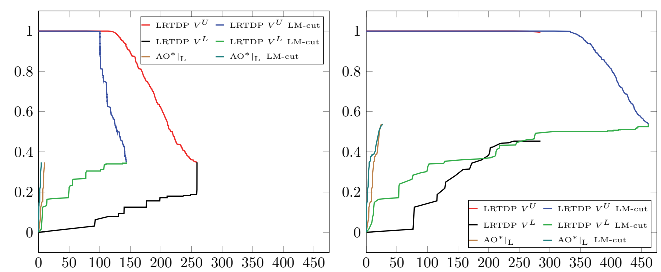

Figure 4: Anytime behavior in LRTDP$|_\mathrm{LU}$ (V^U and V^L) and AO*$|_\mathrm{L}$ (V^L only), as a function of runtime. Elevators instance 11, without pruning and with LM-cut pruning, for constrainedness levels $C = 1.4$ (left) and $C = 1.8$ (right).

As both parts of Figure 4 use the same base instance but with different constrainedness levels C, we can also draw conclusions on the effect of surplus budget. With more budget, more actions can be applied before reaching absorbing states. This adversely affects the upper bound (consistently across our experiments), which takes a much longer time to decrease (cf. $C = 1.8$ vs. $C = 1.4$ in Figure 4). The lower bound, on the other hand, often increases more quickly with higher C as it is easier to find goal states.

Cyclic Planning with FRET

We now consider cyclic planning, pertaining to the standard IPPC benchmarks, and to probabilistic NoMystery, Rovers, TPP without budget (nor resource-) limit. We run only LRTDP, as AO* is restricted to acyclic state spaces. We use the two different variants of FRET described earlier: FRET-V^U as per Kolobov et al. (2011), and our new variant FRET-π^U. We consider all 3 objectives, and the same 4 dead-end pruning methods (as LM-cut returns ∞ iff the cheaper heuristic $h^{\max}$ does, we use $h^{\max}$here).

Domain	#	VI –	VI h^{max}	VI M&S N	VI M&S ∞	VI on BS	FRET-V^U –	FRET-V^U h^{max}	FRET-V^U M&S N	FRET-V^U M&S ∞	FRET-V^U on BS	FRET-π^U –	FRET-π^U h^{max}	FRET-π^U M&S N	FRET-π^U M&S ∞	FRET-π^U on BS
IPPC Benchmarks																
Blocksworld	15	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
Boxworld	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Drive	15	**15**	**15**	**15**	6	6	**15**	**15**	**15**	6	6	**15**	**15**	**15**	6	6
Elevators	15	**15**	**15**	**15**	5	5	**15**	**15**	**15**	5	5	**15**	**15**	**15**	5	5
ExplodingBlocks	15	4	6	4	4	4	4	6	4	4	4	5	**14**	5	4	4
Random	15	0	0	0	0	0	0	0	0	0	0	**4**	**4**	0	0	0
RectangleTireworld	14	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**	**14**
Tireworld	15	10	10	10	10	11	10	11	10	10	11	**15**	**15**	**15**	12	11
Zenotravel	15	**3**	**3**	**3**	1	0	**3**	**3**	**3**	1	1	**3**	**3**	**3**	1	1
Probabilistic Resource-Constrained Benchmarks																
NoMystery	10	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	4	4	4	4	1
Rovers	10	5	5	5	5	**9**	5	5	5	5	**9**	**9**	**9**	**9**	8	**9**
TPP	10	6	6	6	6	6	6	6	6	6	6	**8**	**8**	**8**	6	6
$\sum$	164	81	83	81	60	64	81	84	81	60	65	96	**105**	92	64	61

Table 2: MaxProb coverage in cyclic planning. Best values in **boldface**. FRET-V^U is as per Kolobov et al. (2011), FRET-π^U is our modified version. Both use LRTDP$|_{\text{U}}$. Dead-end pruning variants: "–" none, else based on heuristic value ∞, for h^{max} respectively merge-and-shrink ("N" size bound $N = 100k$, "∞" no size bound). "on BS": run on reduced (bisimulated) state space.

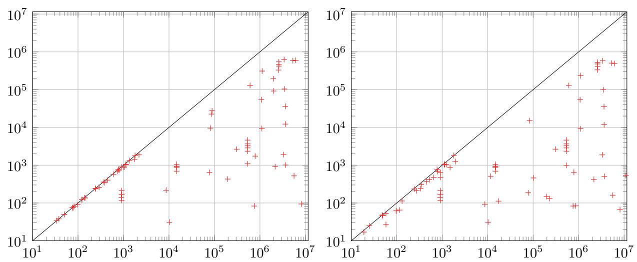

Figure 5: Number of states visited, for VI (x) vs. FRET-π^U (y), with no pruning (left) and with h^{max} pruning (right).

Table 2 shows coverage data for MaxProb. FRET-π^U outperforms both VI and FRET-V^U substantially. Note that, in all domains except ExplodingBlocks and Rovers, the advantage over VI is obtained even without dead-end pruning, i. e., for trivial initialization of V^U. This strongly confirms the power of heuristic search even in the absence of good admissible goal probability estimators. Figure 5 compares search space sizes. Initialization using h^{max} is useful, but gains of 3 orders of magnitude are possible even without it.

The search space size gains in FRET-π^U result in similar runtime gains (see the TR). The single exception is NoMystery, where VI and FRET-V^U visit the same states and FRET-π^U visits ca. 5 times less states, but the larger number of FRET-π^U iterations results in worse runtime, ca. 20 times compared to VI and ca. 5 times compared to FRET-V^U.

With respect to Kolobov et al.'s (2011) experiments on ExplodingBlocks – the previous empirical state of the art – we observe (see TR for details): VI with h^{max} pruning performs similarly to FRET-V^U with h^{max} pruning, showing that Kolobov et al.'s choice to not use pruning in VI indeed obfuscates the possible conclusions; Kolobov et al.'s FRET-V^U also performs similarly, except on the largest instances where SixthSense detects more dead-ends yet this is more than outweighed by the larger runtime overhead; FRET-π^U convincingly outperforms all other algorithms.

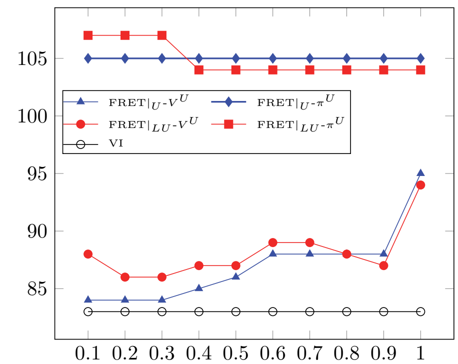

Figure 6: AtLeastProb coverage in cyclic planning, as a function of θ, using h^{max} pruning and default node selection.

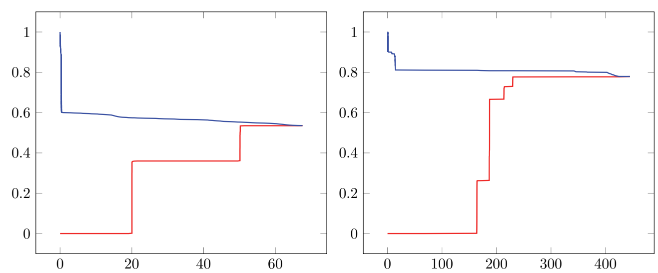

Figure 7: Anytime behavior of FRET-π^U with LRTDP$|_{\text{LU}}$, without pruning for ExplodingBlocks instance 04 (left), and with h^{max} pruning for instance 15 (right).

For the weaker objectives AtLeastProb and ApproxProb, as before we examine coverage as a function of θ respectively δ. Figure 6 shows the data for default node selection in AtLeastProb (the behavior for ApproxProb is qualitatively similar). By FRET$|_{\text{U}}$ respectively FRET$|_{\text{LU}}$, we refer to FRET using LRTDP$|_{\text{U}}$ respectively LRTDP$|_{\text{LU}}$.

For FRET-V^U, the picture is similar to Figure 3 (a), FRET$|_{\text{LU}}$-V^U exhibiting an easy-hard-easy pattern due to the advantages of early termination. For FRET-π^U, though, the curves are flat over θ. This is due to benchmark scaling: in each domain, there is an instance number x so that, below x, FRET-π^U can solve all instances completely (solving MaxProb), while above x neither $V^L(I)$ nor $V^U(I)$ can be improved at all, up to the time/memory limit. On smaller instances, we do get the expected anytime behavior. Figure 7 exemplifies this. The easy-hard-easy pattern would thus emerge for smaller runtime/memory limits.

Conclusion

Optimal goal probability analysis is a notoriously hard problem, to the extent that the amount of work addressing it is limited. We clarified the empirical state of the art, and substantially improved it through a novel variant of FRET and through a novel state-space reduction method. We showed that there are opportunities arising from naturally acyclic problems, and from early termination on criteria weaker than maximum goal probability. We hope that this will inspire renewed interest in this important problem. Promising future directions include advanced admissible goal probability estimators, e. g. from abstractions interpreted as bounded-parameter MDPs (Givan, Leach, and Dean 2000); hybrids of heuristic search with Monte-Carlo tree search, geared at good anytime behavior and thus early termination; and the exploitation of goal probability monotonicity as a function of remaining budget. Simulated pentesting is an application worth algorithms research in it own right. Partial-order reduction appears especially promising there.

Acknowledgments. This work was partially supported by the German Research Foundation (DFG), under grant HO 2169/5-1. We thank Christian Muise for his Probabilistic-PDDL extension of the FD parser. We thank Andrey Kolobov for discussions. We thank the anonymous reviewers, whose comments helped to improve the paper.

References

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

Barto, A. G.; Bradtke, S. J.; and Singh, S. P. 1995. Learning to act using real-time dynamic programming. *AIJ*.

Bertsekas, D. 1995. *Dynamic Programming and Optimal Control, (2 Volumes)*. Athena Scientific.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *AIJ*.

Bonet, B., and Geffner, H. 2003. Labeled RTDP: Improving the convergence of real-time dynamic programming. *ICAPS'03*.

Bonet, B., and Geffner, H. 2006. Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. *ICAPS'06*.

Bryce, D., and Buffet, O. 2008. 6th international planning competition: Uncertainty part. *Proc. IPC'08*.

Camacho, A.; Muise, C.; and McIlraith, S. A. 2016. From FOND to robust probabilistic planning: Computing compact policies that bypass avoidable deadends. *ICAPS'16*.

Coles, A. J.; Coles, A.; García Olaya, A.; Jiménez, S.; Linares López, C.; Sanner, S.; and Yoon, S. 2012. A survey of the 7th international planning competition. *AI Magazine*.

Coles, A. J. 2012. Opportunistic branched plans to maximise utility in the presence of resource uncertainty. *ECAI'12*.

Dai, P.; Mausam; Weld, D. S.; and Goldsmith, J. 2011. Topological value iteration algorithms. *JAIR*.

Dean, T. L., and Givan, R. 1997. Model minimization in markov decision processes. *AAAI'97*.

Domshlak, C., and Mirkis, V. 2015. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *JAIR*.

Givan, R.; Leach, S. M.; and Dean, T. 2000. Bounded-parameter Markov decision processes. *AIJ*.

Hansen, E. A., and Zilberstein, S. 2001. LAO*: a heuristic search algorithm that finds solutions with loops. *AIJ*.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? *ICAPS'09*.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & shrink abstraction: A method for generating lower bounds in factored state spaces. *JACM*.

Helmert, M. 2006. The Fast Downward planning system. *JAIR*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR*.

Hoffmann, J. 2015. Simulated penetration testing: From "Dijkstra" to "Turing Test++". *ICAPS'15*.

Hou, P.; Yeoh, W.; and Varakantham, P. 2014. Revisiting risk-sensitive MDPs: New algorithms and results. *ICAPS'14*.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. *IJCAI'09*.

Kolobov, A.; Mausam; Weld, D. S.; and Geffner, H. 2011. Heuristic search for generalized stochastic shortest path MDPs. *ICAPS'11*.

Kolobov, A.; Mausam; and Weld, D. S. 2010. Sixthsense: Fast and reliable recognition of dead ends in MDPs. *AAAI'10*.

Kolobov, A.; Mausam; and Weld, D. S. 2012. A theory of goal-oriented MDPs with dead ends. *UAI'12*.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Robotics: Science and Systems IV*.

Little, I.; Aberdeen, D.; and Thiébaux, S. 2005. Prottle: A probabilistic temporal planner. *AAAI'05*.

Little, I., and Thiebaux, S. 2007. Probabilistic planning vs replanning. *ICAPS Workshop on the IPC*.

Marecki, J., and Tambe, M. 2008. Towards faster planning with continuous resources in stochastic domains. *AAAI-08*.

McMahan, H. B.; Likhachev, M.; and Gordon, G. J. 2005. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. *ICML'05*.

Meuleau, N.; Benazera, E.; Brafman, R. I.; Hansen, E. A.; and Mausam, M. 2009. A heuristic search approach to planning with continuous resources in stochastic domains. *JAIR*.

Muise, C. J.; McIlraith, S. A.; and Beck, J. C. 2012. Improved non-deterministic planning by exploiting state relevance. *ICAPS'12*.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-constrained planning: A monte carlo random walk approach. *ICAPS'12*.

Sarraute, C.; Buffet, O.; and Hoffmann, J. 2012. POMDPs make better hackers: Accounting for uncertainty in penetration testing. *AAAI'12*.

Smith, T., and Simmons, R. G. 2006. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. *AAAI'06*.

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Revisiting goal probability analysis in probabilistic planning (technical report). Saarland University. http://fai.cs.uni-saarland.de/hoffmann/papers/icaps16a-tr.pdf.

Teichteil-Königsbuch, F.; Kuter, U.; and Infantes, G. 2010. Incremental plan aggregation for generating policies in MDPs. *AAMAS'10*.

Teichteil-Königsbuch, F. 2012. Stochastic safest and shortest path problems. *AAAI'12*.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: a baseline for probabilistic planning. *ICAPS'07*.

Younes, H. L. S.; Littman, M. L.; Weissman, D.; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *JAIR*.