Heuristic Search Planning With Multi-Objective Probabilistic LTL Constraints

Peter Baumgartner, Sylvie Thiébaux, and Felipe Trevizan
Data61/CSIRO and The Australian National University
Email: first.last@anu.edu.au

Abstract

We present an algorithm for computing cost-optimal stochastic policies for Stochastic Shortest Path problems (SSPs) subject to multi-objective PLTL constraints, i.e., conjunctions of probabilistic LTL formulas. Established algorithms capable of solving this problem typically stem from the area of probabilistic verification, and struggle with the large state spaces and constraint types found in automated planning. Our approach differs in two crucial ways. Firstly it operates entirely on-the-fly, bypassing the expensive construction of Rabin automata for the formulas and their prohibitive prior synchronisation with the full state space of the SSP. Secondly, it extends recent heuristic search algorithms and admissible heuristics for cost-constrained SSPs, to enable pruning regions made infeasible by the PLTL constraints. We prove our algorithm correct and optimal, and demonstrate encouraging scalability results.

Introduction

The problem of computing optimal but safe policies for autonomous agents operating in uncertain environments has recently attracted significant attention from the fields of automated verification (Kwiatkowska and Parker 2013), robotics (Ding et al. 2014), and artificial intelligence (Sprauel, Kolobov, and Teichteil-Königsbuch 2014). Such policies must minimise the agent's expected cost to reach a goal, whilst providing probabilistic guarantees about the sequence of visited states. For instance, a policy for a search and rescue UAV mission might need to minimise the expected time to get survivors to safety, whilst avoiding dangerous areas at all times, and circling the affected locations to correctly determine the presence of survivors with high probability.

Such planning problems can be modelled as stochastic shortest path problems (SSPs) augmented with multi-objective probabilistic LTL (MO-PLTL) constraints. Such constraints are conjunctions $\phi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\, \psi_i$ of *objectives*, where ψ_i is an LTL formula and $z_i \subseteq [0, 1]$ is an interval bounding its probability. An optimal solution takes the form of a finite-memory stochastic policy π whose execution: (a) satisfies the MO-PLTL constraints; (b) reaches a goal state with probability 1; and (c) has minimal expected cost, subject to (a,b).

Fulfilling requirement (a) alone, and to a lesser degree in combination with (c), has been studied intensively in the area of policy synthesis for Markov Decision Processes (MDPs) (Forejt et al. 2011; Etessami et al. 2008), leading to what we call the static approach. This consists in constructing k automata whose accepting runs correspond exactly to the satisfying runs of the k LTL constraints, and then building the k+1-ary cross-product of these k automata with the state space of the decision process. As these automata have to be deterministic for the cross product to remain an MDP, deterministic Rabin automata (DRA) are required, which are obtained from nondeterministic Büchi automata (NBA) for the formulas ψ_i. Finally, linear programming is applied to the cross-product, to generate (cost-optimal) policies that reach certain sets of states (bottom-end components) with given probability bounds of the form $> u_i$ or $\geq u_i$, where $u_i \in [0, 1]$. A policy for the original problem can be recovered from the solution of the LP.

There are two practical issues that make the static approach inapplicable to the probabilistic planning problems we are interested in. Firstly, the DRA compilation can be prohibitive for certain common type of planning objectives, e.g., multiple maintenance objectives. Secondly and more significantly, the explicit construction of the cross-product is completely infeasible for planning problems which usually have huge state spaces.

In this paper, we resolve these two issues in the particular case where the goal requirement (b) is present by leveraging and extending efficient heuristic search algorithms and heuristics from the field of constrained SSPs (C-SSPs) (Trevizan et al. 2016; Trevizan, Thiébaux, and Haslum 2017). Specifically, the recent *i-dual* algorithm solves C-SSPs *on-the-fly*, by running linear programming on small state space fragments of increasing size, guided by an admissible heuristic function to prune regions that are too costly or cannot satisfy the constraints. When guided by effective C-SSP heuristics, such as the *occupation measure* heuristics (Trevizan, Thiébaux, and Haslum 2017), i-dual only expands a fraction of the state space. The *i^2-dual* algorithm additionally embeds the computation of occupation measure heuristics in the LP, making optimisation and heuristic estimation synergic and avoiding repeated calls to heuristic estimators. These algorithms and heuristics are however currently limited to much simpler constraint types and do not handle MO-PLTL constraints.

To harness the benefits of this "on-the-fly approach" for our problem, we must (1) bypass the need for computing the DRA and the cross product, and (2) extend the heuristics and algorithm to handle MO-PLTL constraints. We achieve (1) by embedding on-the-fly progression of LTL formulae (Bacchus and Kabanza 1998) or determinisation of NBA in the state space expansion performed by i^2-dual. This avoids doing work that is not relevant to the traces expanded by the heuristic search and, in the case of progression, leads to improved worst-case complexity. We achieve (2) by computing occupation measure heuristics from the exponentially smaller NBAs for each of the formulas, and optionally embedding this computation into i^2-dual. Our experiments show that the on-the-fly approach is capable of solving planning problems that are out of reach of the state-of-the-art implementation of the static approach in the Prism probabilistic model-checker (Kwiatkowska, Norman, and Parker 2011).

Due to the lack of space, proofs of theorems are given in appendix.

Background and Problem Definition

Let $Dist(\mathrm{X})$ be the set of all distributions on a set X. Given a finite set S of states, we write S^+ for the set of (non-empty) finite sequences of states over S, and S^ω for the set of infinite state sequences, thus $\mathrm{S}^+ \cap \mathrm{S}^\omega = \emptyset$. For a state sequence $p \in \mathrm{S}^+ \cup \mathrm{S}^\omega$ and a natural number i, p_i denotes the state of index i in p, and $p(i)$ the suffix $p_i p_{i+1} \cdots$ of p. For a finite sequence $p \in \mathrm{S}^+$, $\mathrm{last}(p)$ represents the last state of p. We write $p; p'$ for the concatenation of $p \in \mathrm{S}^+$ and $p' \in \mathrm{S}^+ \cup \mathrm{S}^\omega$.

SSPs. A *Stochastic Shortest Path problem* (SSP) is a tuple $\mathcal{S} = (\mathrm{S}, s_{\text{init}}, \mathrm{G}, \mathrm{A}, \mathrm{P}, \mathrm{C}, \mathrm{T})$ where: S is the finite set of states; $s_{\text{init}} \in \mathrm{S} \setminus \mathrm{G}$ is the initial state; $\mathrm{G} \subset \mathrm{S}$ is the non-empty set of goal states; A is a finite set of *actions* and $\mathrm{A}(s) \subseteq \mathrm{A}$ is the *set of actions enabled* in $s \in \mathrm{S}$. We assume $\mathrm{A}(s) \neq \emptyset$ for $s \in \mathrm{S} \setminus \mathrm{G}$ and $\mathrm{A}(s_g) = \emptyset$ for $s_g \in \mathrm{G}$; $\mathrm{P}(\bullet|s, \alpha) \in Dist(\mathrm{S})$ is the probability of transitioning to $t \in \mathrm{S}$ after applying $\alpha \in \mathrm{A}(s)$ in state s; $\mathrm{C}(\alpha) \in \mathbb{R}^*_+$ is the *cost* of α; and $\mathrm{T}\colon \mathrm{G} \to \mathbb{R}$ is the one-time *terminal cost* of reaching a goal state.[1]

Policies. A solution to an SSP is a policy. In this paper, we consider various types of policies for SSPs and for more complex sequential decision problems to be defined later. These policy types differ in their ability to incorporate memory and randomisation. The most general of these are *stochastic history-dependent policies*, or *unrestricted policies*, which are (partial) functions $\pi : \mathrm{S}^+ \mapsto Dist(\mathrm{A})$ mapping the finite state history p to a probability distribution over the actions enabled in the current state. We abbreviate $\pi(p)(\alpha)$ as $\pi(p, \alpha)$ and stipulate that $\pi(p, \alpha) = 0$ if $\alpha \notin \mathrm{A}(\mathrm{last}(p))$. A policy is *deterministic* if, for every p in its domain, there is $\alpha \in \mathrm{A}(\mathrm{last}(p))$ such that $\pi(p, \alpha) = 1$; hence these policies are (partial) functions $\pi : \mathrm{S}^+ \mapsto \mathrm{A}$ and we write $\pi(p)$ for the unique action prescribed by the policy in a given state history. A policy is *memoryless* (i.e., stationary) if $\pi(p)$ only depends on $\mathrm{last}(p)$, and has *finite memory* if it additionally depends on a mode which has finitely many values and is updated along with the current state after each transition.

A *run* r is a path $p = s_1 s_2 \cdots \in \mathrm{S}^+ \cup \mathrm{S}^\omega$ annotated with the actions executed between states, that is, $r = s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} s_3 \cdots$ such that $\alpha_i \in \mathrm{A}(s_i)$ and $\mathrm{P}(s_{i+1}|s_i, \alpha_i) > 0$, for all $i \geq 1$. The *cost* and *probability* of a run r are defined as $\mathrm{C}(r) = \sum_{i \geq 1} \mathrm{C}(\alpha_i)$ and $\mathrm{P}(r) = \prod_{i \geq 1} \mathrm{P}(s_{i+1}|s_i, \alpha_i)$, respectively. Since a path p can be trivially obtained from a run r by removing the action annotation, we use r as a path (e.g., $\pi(r, a)$) when clear from context.

Given a policy π, a run r is an *exhaustive run of* π if $\pi(s_1 \cdots s_i, \alpha_i) > 0$ for all $i \geq 1$ and either r is infinite or π is not defined for the finite path represented by r. An exhaustive run of π only stops when π is unable to recommend an action to be executed next (in particular when a goal state is reached). We write $\mathrm{Runs}(s, \pi)$ for the *set of all exhaustive runs of* π *starting from* $s \in \mathrm{S}$, and $\mathrm{GRuns}(s, \pi) \subseteq \mathrm{Runs}(s, \pi)$ for those runs that additionally reach a goal state. For any $s \in \mathrm{S}$ and $r \in \mathrm{Runs}(s, \pi)$, the *probability of* r *being produced by* π is $\mathrm{P}(r|\pi) = \mathrm{P}(r) \prod_{i \geq 1} \pi(s_1 \cdots s_i, \alpha_i)$.

A policy π is *proper* if $\sum_{r \in \mathrm{GRuns}(s_{\text{init}}, \pi)} \mathrm{P}(r|\pi) = 1$, i.e., if the probability of reaching the goal when using π from s_{init} is 1. To simplify notation, we assume that there is at least one proper policy for $\mathcal{S}$, or equivalently, that there are no reachable dead ends from s_{init}. Dead ends can easily be handled as in (Trevizan, Teichteil-Königsbuch, and Thiébaux 2017).

Optimal Policies. Given a proper policy π, its *total expected cost* $\mathrm{V}^\pi(s)$ to reach a goal state from a state $s \in \mathrm{S}$ is:

$$\mathrm{V}^\pi(s) = \sum_{r \in \mathrm{GRuns}(s,\pi)} \Big[\mathrm{C}(r) + \mathrm{T}(\mathrm{last}(r))\Big] \mathrm{P}(r|\pi).$$

We abbreviate $\mathrm{V}^\pi(s_{\text{init}})$ with V^π. An *optimal policy* π^* is a proper policy such that $\mathrm{V}^{\pi^*} \leq \mathrm{V}^\pi$ for all proper policies π.

It is well-known that, in the case of SSPs, at least one optimal policy is memoryless and deterministic. For such policies, their total expected cost can be expressed as the following set of fixed-point equations that is at the core of most solution methods for SSPs:

$$\mathrm{V}^\pi(s) = \begin{cases} \mathrm{T}(s) & \text{if } s \in \mathrm{G} \\ \mathrm{C}(\pi(s)) + \sum_{t \in \mathrm{S}} \mathrm{P}(t|s, \pi(s)) \mathrm{V}^\pi(t) & \text{otherwise} \end{cases}$$

C-SSPs. Many extensions of SSPs require stochastic policies. This is the case of *cost-constrained SSPs* (C-SSPs), which will be useful later in this paper, and for which stochastic policies are needed to optimally account for trade-offs between various cost functions representing, e.g., fuel, money, or time. A C-SSP $\mathcal{C} = (\mathrm{S}, s_{\text{init}}, \mathrm{G}, \mathrm{A}, \mathrm{P}, \mathbf{C}, \mathbf{T}, \boldsymbol{z})$ is an SSP where: (i) the action cost function is replaced by a vector of $k+1$ action cost functions $\mathbf{C} = [\mathrm{C}_0, \ldots, \mathrm{C}_k]$ ($\mathrm{C}_0\colon \mathrm{A} \to \mathbb{R}^*_+$ and $\mathrm{C}_j : \mathrm{A} \to \mathbb{R}_+$ for all $j \geq 1$); (ii) the terminal cost function is also replaced by a vector of $k+1$ terminal cost functions $\mathbf{T}$; and (iii) a vector of k intervals $\boldsymbol{z} = [z_1, \ldots, z_k]$ ($z_j \subseteq \mathbb{R}_+$

[1] An SSP with terminal cost can be trivially encoded as an SSP without terminal costs by adding extra actions. We use terminal costs to simplify our notation.

for all $j \geq 1$) is added. We refer to C_0 and T_0 as the *primary* action (resp. terminal) cost and to the other elements of the cost vectors as the *secondary* costs. The optimal solution for a C-SSP is any stochastic memoryless policy $\pi : S \mapsto Dist(A)$ which minimises the total expected primary cost to reach a goal state in G from the initial state s_{init} subject to the total expected j-th cost lying within interval z_j for $j \geq 1$. That is, there are $k+1$ total expected cost functions:

$$V_j^\pi(s) = \begin{cases} T_j(s) & \text{if } s \in G \\ \sum_{\alpha \in A(s)} \pi(s,\alpha)\left(C_j(\alpha) + \sum_{t \in S} P(t|s,\alpha) V_j^\pi(t)\right) & \text{otherwise} \end{cases} \quad (1)$$

and an optimal policy π^* minimizes $V_0^{\pi^*}$ subject to $V_j^{\pi^*} \in z_j$, for all $j \geq 1$.

Algorithms. The worst-case complexity of computing an optimal policy for SSPs or C-SSPs is polynomial in the size of the state space S (Dolgov and Durfee 2005). For planning problems however, the state space is much too large to be explicitly enumerated. Therefore, the field represents SSPs using exponentially more compact factored representations, and has moved away from methods that completely expand the state space, including in particular from vanilla linear programming and dynamic programming methods such as value and policy iteration. Instead it focuses on Monte Carlo tree search (Bonet and Geffner 2012; Keller and Eyerich 2012), or on heuristic search approaches such as (L)RTDP (Barto, Bradtke, and Singh 1995; Bonet and Geffner 2003) and LAO* (Hansen and Zilberstein 2001) for SSPs, and i-dual for C-SSPs (Trevizan et al. 2016). These start from the factored representation and expand the state space on-the-fly, guided by an *admissible heuristic* function derived from a polynomial time analysis of the factored representation (Bonet and Geffner 2005; Teichteil-Königsbuch, Vidal, and Infantes 2011; Trevizan, Thiébaux, and Haslum 2017). Such a heuristic provides us with a lower bound on the optimal expected cost $V^{\pi^*}(s)$ to reach the goal from the current state s, and enables large regions of the state space to be pruned. When equipped with an informative heuristic, heuristic search algorithms often expand only a small fraction of the state space.

Factored Representation. We adopt a probabilistic variant of the SAS$^+$ formalism as our factored representation (Backström 1992; Trevizan, Thiébaux, and Haslum 2017). A *probabilistic SAS$^+$ task* is a tuple $\langle \mathcal{V}, A, s_\bullet, s_\star, C \rangle$. $\mathcal{V}$ is a finite set of *state variables*, and each $v \in \mathcal{V}$ has a finite domain D_v. A *valuation* (or partial state) is a function s on a subset $\mathcal{V}_s$ of $\mathcal{V}$, such that $s[v] \in D_v$ for $v \in \mathcal{V}_s$ and $v = \bot$ otherwise. If $\mathcal{V}_s = \mathcal{V}$, then s is a *state*. $s_\bullet$ is the initial state and $s_\star$ is a partial state representing the goal. Given partial states s and s', we write $s' \subseteq s$ if $s'[v] = s[v]$ for all $v \in \mathcal{V}_{s'}$.

The *result* of applying a valuation e to valuation s is the valuation $res(s,e)$ 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 $\alpha \in A$ consists of a *precondition* $pre(\alpha)$ given by a valuation over $\mathcal{V}$, a set $\mathit{eff}(\alpha)$ of *effects*, each of which is a valuation over $\mathcal{V}$, and a probability distribution $\Pr_\alpha(\cdot) \in Dist(\mathit{eff}(\alpha))$ such that $\Pr_\alpha(e)$ represents the probability of $res(s,e)$ being the state resulting from applying α in state s. $C(\alpha) \in \mathbb{R}_+^*$ is the immediate cost of applying α.

A probabilistic SAS$^+$ task $\langle \mathcal{V}, A, s_\bullet, s_\star, C \rangle$ defines an SSP $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ where $s_{\text{init}} = s_\bullet$, $S = \bigtimes_{v \in \mathcal{V}} D_v$, $G = \{s \in S | s_\star \subseteq s\}$, $A(s) = \{\alpha \in A | pre(\alpha) \subseteq s\}$, $P(t|s,\alpha) = \sum_{e \in \mathit{eff}(\alpha) \text{ s.t. } t = res(s,e)} \Pr_\alpha(e)$, and $T(s) = 0$ for all states $s \in G$. A C-SSP can be compactly represented by a probabilistic SAS$^+$ task whose cost function has been replaced with the corresponding vectors of cost functions and intervals.

SSPs with Multi-Objective PLTL Constraints

The main contribution of this paper is to extend the scope of heuristic search to SSPs with multi-objective probabilistic LTL constraints, which we define next.

LTL. Let $\langle \mathcal{V}, A, s_\bullet, s_\star, C \rangle$ be a probabilistic SAS$^+$ task and $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ the SSP it defines. Let $AP = \{(v,d) \mid v \in \mathcal{V}, d \in D_v\}$ be the finite set of *atoms*. LTL formulas ψ are described by the following grammar:

$$\psi = \mathsf{true} \mid (v,d) \in AP \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi \mid \mathbf{X}\,\psi \mid \psi\,\mathbf{U}\,\psi$$

The standard semantics of LTL interprets formulas over infinite sequences of states. Formally, for a formula ψ and a path $p \in S^\omega$, the satisfaction relation $p \models \psi$ is defined as follows:

$p \models \top$ $\qquad$ $p \models \psi_1 \wedge \psi_2$ iff $p \models \psi_1$ and $p \models \psi_2$

$p \models (v,d)$ iff $p_1[v] = d$ $\qquad$ $p \models \psi_1 \vee \psi_2$ iff $p \models \psi_1$ or $p \models \psi_2$

$p \models \neg\psi$ iff $p \not\models \psi$ $\qquad$ $p \models \mathbf{X}\,\psi$ iff $p(2) \models \psi$

$p \models \psi_1\,\mathbf{U}\,\psi_2$ iff $\exists i \geq 1$ s.t. $p(i) \models \psi_2$ and $p(j) \models \psi_1\ \forall 1 \leq j < i$

Planning, however, seeks finite state sequences ending in a goal state, and therefore usually interprets LTL over *finite* paths. A popular semantics for that is the *infinite extension semantics* (Bacchus and Kabanza 1998; Bauer and Haslum 2010), which gives a formula the truth value it would have under the standard semantics by infinite repetition of a path's last state. (Other popular semantics include f-FOLTL (Baier and McIlraith 2006) and LTL$_f$ (De Giacomo and Vardi 2013), but they do not appear to offer any advantage in our context.) That is, for any $p \in S^+$, we define the satisfaction relation $p \models_{\text{IE}} \psi$ as follows:

$$p \models_{\text{IE}} \psi := \begin{cases} \mathsf{true} & \text{if } p; (\text{last}(p))^\omega \models \psi \\ \bot & \text{Otherwise} \end{cases}$$

PLTL. A *probabilistic LTL (PLTL) formula* is of the form $\mathbf{P}_{\in z}\,\psi$ where $z \subseteq [0,1]$. Informally, $\mathbf{P}_{\in z}\,\psi$ states that the LTL formula ψ holds with a probability that lies in the interval z. A *multi-objective PLTL (MO-PLTL) formula* is a conjunction $\phi = \bigwedge_{i=1}^k \mathbf{P}_{\in z_i}\,\psi_i$ of PLTL formulas, for some *arity* $k \geq 0$. If $k = 0$ then ϕ is equivalent to the constant true.

The probability of ψ being satisfied by $\mathcal{S}$ under π is defined as $\Pr_{\mathcal{S}}^\pi(\psi) = \sum_{r \in GRuns(s_{\text{init}},\pi) \text{ s.t. } r \models_{\text{IE}} \psi} P(r|\pi)$. We say that $\mathcal{S}$ and π *satisfy* $\phi = \bigwedge_{i=1}^k \mathbf{P}_{\in z_i}\,\psi_i$ and write $\mathcal{S}, \pi \models \phi$ iff $\Pr_{\mathcal{S}}^\pi(\psi_i) \in z_i$ for all $i = 1..k$.

We can now state the problem we want to solve:

Definition 1 (MO-PLTL SSP Problem) *Let $\mathcal{T}$ be a probabilistic SAS$^+$ task, $\mathcal{S}$ the SSP it defines, and ϕ an MO-PLTL formula. Find an optimal policy π^* for $\mathcal{S}$ and ϕ, i.e., $\mathcal{S}, \pi^* \models \phi$ and $V^{\pi^*} \leq V^{\pi}$ for all unrestricted proper policies π for $\mathcal{S}$ such that $\mathcal{S}, \pi \models \phi$. Return failure if no unrestricted proper policy π for $\mathcal{S}$ exists such that $\mathcal{S}, \pi \models \phi$.*

Related Work

Algorithms that can be used for the MO-PLTL SSP problem (Def. 1) are given in (Kwiatkowska and Parker 2013; Forejt et al. 2011; Etessami et al. 2008). The overall approach is similar to single-LTL policy synthesis in that it constructs a product automaton and reduces to certain reachability problems determined by the end-components of that automaton. These algorithms can be used for both synthesis and model checking, i.e., to check that all policies satisfy the given MO-PLTL formula ϕ; see, e.g., (Kwiatkowska and Parker 2013). They are based on building NBAs for each LTL formula ψ_i followed by transformation into DRAs. Both steps require, worst-case exponential time and space each, and are together double-exponential in $|\phi|$ (and polynomial in $|S|$). Also, they accept single-sided bounds only, $> u_i$ or $\geq u_i$, where $u_i \in [0, 1]$. Our interval bounds "$\in z_i$" can be compiled away into that form at the cost of doubling the size of ϕ. This can worsen the overall costs, hence, to double-exponential in $2|\phi|$ which can be problematic even for small ϕ.

It is important to note that the mentioned algorithms solve the more general problem of policy synthesis for MDPs as opposed to SSPs, that is, probabilistic problems with no goal states. This problem is known to be complete for double-exponential time (Courcoubetis and Yannakakis 1995). Indeed, we can show that our tailored SSP synthesis algorithm is strictly less complex (Theorem 4) and no extra penalty needs to be paid for accepting intervals.

It could be argued that MO-PLTL in the context of SSPs is closer to synthesis for probabilistic LTL over *finite* traces. Indeed, automata-based approaches to planning problems for LTL constraints over finite trace semantics have been proposed in (De Giacomo and Vardi 2013; 2015). These techniques could be employed for MO-PLTL policy synthesis following the architecture outlined above (cf. (Kwiatkowska and Parker 2013)). However, this would again incur double-exponential worst-time costs, for building NFAs and determinising them afterwards.

In yet another approach (Lacerda, Parker, and Hawes 2015; 2017) consider syntactic co-safe MO-PLTL constraints, which are reachability queries and can be represented by DFAs of double-exponential size in the size of the contraints. Notice that our MO-PLTL constraints are not restricted in that way and allow for, e.g., the formulation of *maintenance goals* (see below for examples).

The existence of goals make MO-PLTL SSPs amenable (also) to heuristic search that builds only a small fraction of the state space on-the-fly. Our approach extends existing work from the planning community on deterministic and non-deterministic planning with LTL constraints to the probabilistic case. Relevant work include in particular compilation approaches which translate LTL and finite LTL variants into various types of automata whose states, transitions and accepting conditions are then incorporated as additional variables, actions, or constraints in the factored planning problem description (Edelkamp 2006; Baier and McIlraith 2006; Torres and Baier 2015; Camacho et al. 2017). The two key advantages of these approaches and ours are: a) they deal with exponentially more compact non-deterministic automata, leaving to the planner the choice of how to resolve the non-determinism, and b) they use efficient planning algorithms (e.g. planning via heuristic search), which do not explicitly generate the whole state space.

The use of progression as an alternative to automata to generate modes originated in the TLPlan planner (Bacchus and Kabanza 1998). Progression has been used in the probabilistic planning setting, for instance to accommodate non-Markovian rewards (Thiébaux et al. 2006), but not to handle PLTL constraints.

MO-PLTL SSPs as Constrained SSPs

This section contains our formal framework for translating MO-PLTL SSPs into Constrained SSPs (C-SSPs) to be solved for finite-memory policies. The translation abstracts from how MO-PLTL constraints are dealt with. We then describe two instances of this framework which respectively use automata and progression to represent the policy modes.

In this section let $\mathcal{T}$ be a probabilistic SAS$^+$ task, $\mathcal{S}$ the SSP it defines, and $\phi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\, \psi_i$ an MO-PLTL formula.

Finite-Memory Policies. (Baier and Katoen 2008) A *stochastic finite-memory policy for an SSP $\mathcal{S}$* is a DFA $\pi_{\text{fin}} = (M, \text{start}, \text{mod}, \text{act})$ where: M is a finite set of *modes*, $\text{start} \in M$ is an initial mode, $\text{mod}\colon M \times S \mapsto M$ is the mode transition function, and $\text{act}\colon M \times S \mapsto \mathit{Dist}(A)$ is the action probability function such that, for all $\langle m, s\rangle \in M \times S$, $\text{act}(m, s)(\alpha) \geq 0$ only if $\alpha \in A(s)$ and 0 otherwise. A finite-memory policy π_{fin} can be used whenever an unrestricted policy π is required by defining $\pi(s_1 \cdots s_n) = \text{act}(m_n, s_n)$ where $m_1 = \text{start}$ and $m_i = \text{mod}(m_{i-1}, s_i)$ for $i = 2..n$.[2]

Below we define modes M_i and components start_i and mod_i, for $i = 1..k$. These are compounded into a partial finite-memory policy $(\mathbf{M}, \mathbf{start}, \mathbf{mod}, \cdot)$ for $\mathcal{S}$, where $\mathbf{M} = M_1 \times \cdots \times M_k$, $\mathbf{start} = \langle \text{start}_1, \ldots, \text{start}_k\rangle$, and $\mathbf{mod}(\langle m_1, \ldots, m_k\rangle, s) = \langle \text{mod}_1(m_1, s), \ldots, \text{mod}_k(m_k, s)\rangle$. Then we compile into a C-SSP $C^\times$ such that any optimal policy $\pi^*_{C^\times}$ for $C^\times$ can be used as the act-component for the finite-memory policy:

Definition 2 (Product C-SSP $C^\times$) *Given an MO-PLTL SSP $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ and $(\mathbf{M}, \mathbf{start}, \mathbf{mod}, \cdot)$*

[2]The literature usually defines the mode transition function mod dependent on the source state s_{i-1}, not the target state s_i, i.e., $m_i = \text{mod}(m_{i-1}, s_{i-1})$. This change is theoretically inconsequential, but technically more convenient for us.

as above, the product C-SSP $C^\times$ *is the C-SSP* $(S^\times, s^\times_{\text{init}}, G^\times, A, P^\times, \mathbf{C}^\times, \mathbf{T}^\times, z)$ *with k secondary costs where: the state space* $S^\times = \mathbf{M} \times S$; *the initial state* $s^\times_{\text{init}} = \langle \mathbf{start}, s_{\text{init}} \rangle$; *the goal set* $G^\times = \mathbf{M} \times G$; $A(\langle \boldsymbol{m}, s \rangle) = A(s)$ *for all* $\boldsymbol{m} \in \mathbf{M}$; *the transition probability function is* $P^\times(\langle \boldsymbol{n}, t \rangle | \langle \boldsymbol{m}, s \rangle, \alpha) = P(t|s, \alpha)$ *if* $\alpha \in A(s)$ *and* $\boldsymbol{n} = \mathbf{mod}(\boldsymbol{m}, t)$*, otherwise* 0; *the main action cost* $C^\times_0 = C$*, and* $C^\times_i(\alpha) = 0$ *for all* $\alpha \in A$ *and* $i \in 1..k$; $T^\times_0(\bullet) = 0$ *and, for* $i \in 1..k$, $T^\times_i(\langle \boldsymbol{m}, s \rangle) = 1$ *if* $\langle \boldsymbol{m}, s \rangle \in \text{Accept}_i$ *and* 0 *otherwise*; *and* z_i *is the probability interval for* ψ_i *in* ϕ.

The mentioned sets $\text{Accept}_i \subseteq G^\times$, defined below, are mode specific. Informally, $\langle \boldsymbol{m}, s \rangle \in \text{Accept}_i$ means that ψ_i is IE-satisfied by all finite runs of $\mathcal{S}$ from s_{init} to the goal s.

As $C^\times$ is an ordinary C-SSP, any off-the-shelf C-SSP solver can be used to compute an optimal policy $\pi^*_{C^\times}$ for $C^\times$. Moreover, $\pi^*_{C^\times}$ is stationary and stochastic (Altman 1999), that is, $\pi^*_{C^\times}$ maps states $\langle \boldsymbol{m}, s \rangle \in S^\times$ to probability distributions over the set of actions $A(s)$. Thus, $\pi^*_{C^\times}$ can be used as action probability function, i.e., $\text{act}(\boldsymbol{m}, s)(\alpha) = \pi^*_{C^\times}(\langle \boldsymbol{m}, s \rangle, \alpha)$.

Büchi Automaton Mode

In this section we instantiate our C-SSP framework with NBA-based modes. Unlike the policy synthesis methods in "Related Work", we avoid NBA determinisation at up-front exponential costs by using heuristic search and on-the-fly determinisation of the NBA.

In order to employ existing LTL to NBA algorithms we need to equip the given formula with IE-semantics: for an LTL formula ψ let $\psi^{\text{IE}} = \psi \wedge \mathbf{F}\,(\bigwedge_{a \in \text{AP}}(a \rightarrow \mathbf{G}\,a) \wedge (\neg a \rightarrow \mathbf{G}\,\neg a))$, which is a faithful encoding of ψ wrt. the IE-semantics in standard LTL (Bauer and Haslum 2010). Let $\mathcal{B}_{\psi^{\text{IE}}}$ denote an NBA for ψ^{IE}. Then it follows that $\mathcal{B}_{\psi^{\text{IE}}}$ accepts a run r iff $r = p; (\text{last}(p))^\omega$ and $p \models_{\text{IE}} \psi$, for some finite path p.

In more detail, let $\mathcal{B}_{\psi^{\text{IE}}} = (Q, S, \Delta, Q_{\text{init}}, F)$, where: Q is the finite set of states; S is the input alphabet (i.e., the set of states of the given SSP $\mathcal{S}$); $\Delta\colon Q \times S \rightarrow 2^Q$ is the non-deterministic transition function; $Q_{\text{init}} \subseteq Q$ are the initial states; and $F \subseteq Q$ is the acceptance set. As a non-standard notion, we say that a state $q \in Q$ is *accepting with* $s \in S$ iff (1) starting from q some strongly connected component $scc \subseteq Q$ is reachable by 0 or more transitions with s only, (2) $scc \cap F \neq \emptyset$, and (3) $\Delta(q', s') = \emptyset$ for all $q' \in \text{SCC}(q)$ and $s' \in S$ with $s \neq s$.[3] Now we can characterize satisfaction in a pleasant way:

Lemma 3 *Let* $p \in S^+$ *be a path with* $p_1 = s_{\text{init}}$*,* ψ *an LTL formula and* $\mathcal{B}_{\psi^{\text{IE}}}$ *as defined above. Then* $p \models_{\text{IE}} \psi$ *iff starting from some state in* Q_{init} *and following the states in p a state* $q \in Q$ *is reachable that is accepting with* $\text{last}(p)$.

For each LTL formula ψ_i of the given MO-PLTL constraint ϕ we need one NBA, denoted by $\mathcal{B}_{\psi_i^{\text{IE}}} = (Q_i, S, \Delta_i, Q_{\text{init},i}, F_i)$.

[3] In words, condition 3 says that the only transitions among any states in *scc* are with *s*.

The *Büchi-Automata based finite-memory policy (NBA policy)* is $\pi^{\text{NBA}} = (\mathbf{M}^{\text{NBA}}, \mathbf{start}^{\text{NBA}}, \mathbf{mod}^{\text{NBA}}, \pi^*_{C^\times})$ where: $M^{\text{NBA}}_i = 2^{Q_i}$ (the powerset of the NBA state space); $\text{start}^{\text{NBA}}_i = \bigcup_{q \in Q_{\text{init}}} \Delta_i(q, s_{\text{init}})$; $\text{mod}^{\text{NBA}}_i(m, s) = \bigcup_{q \in m} \Delta_i(q, s)$; $\pi^*_{C^\times}$ is an optimal solution for $C^\times$ obtained using $\mathbf{M}^{\text{prog}}, \mathbf{start}^{\text{prog}}, \mathbf{mod}^{\text{prog}}$ and $\text{Accept}_i = \{\langle \boldsymbol{m}, s \rangle \in G^\times \mid \text{some } q \in m_i \text{ is accepting with } s\}$.

To save space we do not spell out soundness and completeness results. They are analogous to Theorems 5 and 6 below. The main difference is in the definition of Accept_i, which, as per Lemma 3 correctly identifies IE-satisfaction of ψ_i.

Formula Progression Mode

By $\text{CNF}(\psi, s)$ we denote the conversion of the LTL formula ψ into conjunctive normal form with on-the-fly simplification by evaluating non-temporal formulas in $s \in S$. Thanks to simplification and equivalences like $\psi_1 \,\mathbf{U}\, \psi_2 \equiv \psi_2 \vee (\psi_1 \wedge \mathbf{X}\,(\psi_1 \,\mathbf{U}\, \psi_2))$, every formula in $\text{CNF}(\psi, s)$ is an $\mathbf{X}$-formula. This allows us to take $\text{CNF}(\psi, s)$ to successor states by means of an un$\mathbf{X}$-operator, which strips each formula in $\text{CNF}(\psi, s)$ of its $\mathbf{X}$-operator. By $\Sigma(\psi_i)$ we denote a certain set of formulas obtained from the i-th PLTL constraint ψ_i (of size quadratic in the size of ψ_i.) See the appendix for details.

The *progression-based finite-memory policy is* $\pi^{\text{prog}} = (\mathbf{M}^{\text{prog}}, \mathbf{start}^{\text{prog}}, \mathbf{mod}^{\text{prog}}, \pi^*_{C^\times})$ where: $M^{\text{prog}}_i = 2^{\Sigma(\psi_i)^3}$; $\text{start}^{\text{prog}}_i = \text{CNF}(\psi_i, s_{\text{init}})$; $\text{mod}^{\text{prog}}_i(m, s) = \text{CNF}(\text{un}\mathbf{X}(m), s)$; and $\pi^*_{C^\times}$ is an optimal solution for $C^\times$ obtained using $\mathbf{M}^{\text{prog}}, \mathbf{start}^{\text{prog}}, \mathbf{mod}^{\text{prog}}$ and $\text{Accept}_i = \{\langle \boldsymbol{m}, s \rangle \in G^\times \mid s \models_{\text{IE}} m_i\}$. Notice that π^{prog} exists iff $C^\times$ has a solution.

Notice that, every mode is a set of sets formulas of size at most 3 over an a priori fixed domain $\Sigma(\psi_i)$ whose size is polynomial in the size of ψ. This is possible thanks to a polynomial "Tseitin-style" CNF transformation (Tseitin 1968).

Theorem 4 (Complexity) *Let* $\mathcal{S}$ *be an SSP,* ϕ *an MO-PLTL constraint, and* $C^\times$ *the C-SSP obtained using* $\mathbf{M}^{\text{prog}}$, $\mathbf{start}^{\text{prog}}$*, and* $\mathbf{mod}^{\text{prog}}$*. Then there is a linear program* $\text{LP}_{C^\times}$ *whose solution, if any, defines the optimal policy* $\pi^*_{C^\times}$ *for* $C^\times$*.* $\text{LP}_{C^\times}$ *can be obtained in time and space at most* $O(|S|) \cdot (2^{O(|\phi|)^7})$.

The mentioned linear program $\text{LP}_{C^\times}$ and the mapping of its solution to the solution of $C^\times$ is defined in the Appendix. We emphasize that Theorem 4 is a marked improvement over using generic model-checking algorithms of double-exponential complexity (see "Related Work"). Our qualitative main results are as follows.

Theorem 5 (Soundness) *If* π^{prog} *exists then it is an optimal policy for* $\mathcal{S}$ *and* ϕ*, i.e.,* $\mathcal{S}, \pi^{\text{prog}} \models \phi$ *and* $V^{\pi^{\text{prog}}} \leq V^\pi$ *for all unrestricted proper policies* π *such that* $\mathcal{S}, \pi \models \phi$.

Theorem 6 (Completeness) *If* π *is an unrestricted proper policy for* $\mathcal{S}$ *such that* $\mathcal{S}, \pi \models \phi$ *then* π^{prog} *exists and is a proper policy such that* $\pi^{\text{prog}} \models \phi$ *and* $V^{\pi^{\text{prog}}} \leq V^\pi$.

Theorems 5 and 6 entail solvability of the MO-PLTL SSP Problem (Def. 1). By Theorem 4, $\mathrm{LP}_{C^\times}$ can be used for that. Finally, we need to add that the complexity results in this section are valid in the context of the progression mode but not when NBAs are used, whether to compute modes or heuristics as we do in the next section.

Heuristic Search Algorithms

In the previous sections, we showed that an MO-PLTL SSP can be compiled into a Constrained SSP (C-SSP) and, in this section, we leverage this result in order to solve MO-PLTL SSPs using heuristic search algorithms for C-SSPs.

Traditionally, C-SSPs are solved as a single LP representing all the (reachable) search space at once similarly to Value Iteration (D'Epenoux 1963; Altman 1999). In the AI community this LP is referred as the dual LP for (C-)SSPs and its variables are the policy's occupation measures $x_{s,a}$ representing the expected number of times action $a \in \mathrm{A}(s)$ will be executed in state s. The main advantage of the dual formulation is that the expectation of any function $f\colon \mathrm{S} \times \mathrm{A} \to \mathbb{R}$ (e.g., the cost functions C_j) over the policy encoded by x can be easily computed by $\sum_{s,a} x_{s,a} f(s,a)$.

Another important feature of the dual formulation that we exploit in this section is that it can be interpreted as a *probabilistic flow problem*, where $x_{s,a}$ describes the flow leaving state s via action a. Using this interpretation, we can see the dual LP as a flow problem where the expected total cost to reach a goal (sink) from the initial state (source) is minimised.

The drawback of this single dual LP approach is the same as that of Value Iteration, namely, the whole reachable search space of the problem must the computed a priori, making this approach not viable for large problems. To address this issue, i-dual (Trevizan et al. 2016) and its successor i^2-dual (Trevizan, Thiébaux, and Haslum 2017) were introduced. Both algorithms solve C-SSPs using heuristic search by generating and solving increasingly large LPs.

i^2-dual for Product C-SSPs

In our context, given a Product C-SSP $C^\times = (\mathrm{S}^\times, s^\times_{\text{init}}, \mathrm{G}^\times, \mathrm{A}, \mathrm{P}^\times, \mathbf{C}^\times, \mathbf{T}^\times, \boldsymbol{z})$ and $\mathcal{T}$ the SAS$^+$ task associated with $C^\times$ as input, i^2-dual incrementally generates and explores larger *partial problems* of $C^\times$ starting from $s^\times_{\text{init}}$. Given a set $\hat{\mathrm{S}} \subseteq \mathrm{S}^\times$ of explored states and a set $\Gamma \subseteq \hat{\mathrm{S}}$ of fringe states of the search, the partial problem solved by i^2-dual is shown in LP1 where, for readability, we abbreviate $\mathrm{in}(\langle \boldsymbol{m}, s\rangle)$ as $\mathrm{in}(\boldsymbol{m}, s)$, $\mathrm{out}(\langle \boldsymbol{m}, s\rangle)$ as $\mathrm{out}(\boldsymbol{m}, s)$ and $x_{\langle \boldsymbol{m},s\rangle,\alpha}$ as $x_{\boldsymbol{m},s,\alpha}$. The constraints of LP1 can be categorized as follows (Fig. 1):

Probabilistic Flow Network (C3–C6). Representation of the states explored so far using occupation measures.

Multiplexer (C7–C8). These constraints extract the flow from Γ and, for each SAS$^+$ variable $v \in \mathcal{V}$, they redirect a copy of this flow to the appropriate state $d \in \mathrm{D}_v$ of *the projection of $\mathcal{T}$ onto v*.

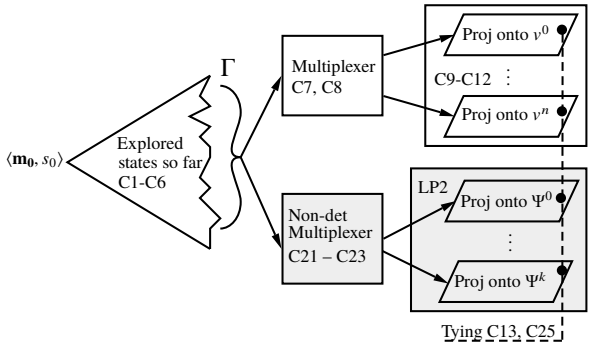

Figure 1: Representation of the flow network solved in each iteration of i^2-dual (non-shaded network – LP1) and of our novel algorithm PLTL-dual (full network – LP3).

$$\min_{x} \sum_{\langle \boldsymbol{m},s\rangle \in \hat{\mathrm{S}}, \alpha \in \mathrm{A}(s)} x_{\boldsymbol{m},s,\alpha} \mathrm{C}^\times_0(\alpha) + \sum_{\langle \boldsymbol{m},s\rangle \in \hat{\mathrm{S}} \cap \mathrm{G}^\times} \mathrm{in}(\boldsymbol{m}, s) \mathrm{T}^\times_0(\langle \boldsymbol{m}, s\rangle) + \sum_{d \in \mathrm{D}_{v'}, \alpha \in \mathrm{A}} x^{v'}_{d,\alpha} \mathrm{C}^\times_0(\alpha) \tag{LP1}$$

$$\text{s.t.} \quad x_{\boldsymbol{m},s,\alpha} \geq 0 \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{\mathrm{S}}, \alpha \in \mathrm{A}(s) \tag{C1}$$

$$x^v_{d,\alpha} \geq 0 \qquad \forall v \in \mathcal{V}, d \in \mathrm{D}_v, \alpha \in \mathrm{A} \tag{C2}$$

$$\mathrm{in}(\boldsymbol{m}, s) = \sum_{\langle \boldsymbol{n},t\rangle \in \hat{\mathrm{S}}, \alpha \in \mathrm{A}(s')} x_{\boldsymbol{n},t,\alpha} \mathrm{P}^\times(\langle \boldsymbol{m}, s\rangle | \langle \boldsymbol{n}, t\rangle, \alpha) \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{\mathrm{S}} \tag{C3}$$

$$\mathrm{out}(\boldsymbol{m}, s) = \sum_{\alpha \in \mathrm{A}(s)} x_{\boldsymbol{m},s,\alpha} \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{\mathrm{S}} \setminus \mathrm{G}^\times \tag{C4}$$

$$\mathrm{out}(\mathbf{start}, s_{\text{init}}) - \mathrm{in}(\mathbf{start}, s_{\text{init}}) = 1 \tag{C5}$$

$$\mathrm{out}(\boldsymbol{m}, s) - \mathrm{in}(\boldsymbol{m}, s) = 0 \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{\mathrm{S}} \setminus \mathrm{G}^\times \tag{C6}$$

$$p^v_0(g) = \sum_{\langle \boldsymbol{m},s\rangle \in \hat{\mathrm{S}} \cap \mathrm{G}^\times} \mathrm{in}(\boldsymbol{m}, s) \qquad \forall v \in \mathcal{V} \tag{C7}$$

$$p^v_0(d) = \sum_{\langle \boldsymbol{m},s\rangle \in \Gamma, s[v]=d} \mathrm{in}(\boldsymbol{m}, s) \qquad \forall v \in \mathcal{V}, d \in \mathrm{D}_v \tag{C8}$$

$$in^v(d) = \sum_{d' \in \mathrm{D}_v, \alpha \in \mathrm{A} \cup \{a_g\}} x^v_{d',\alpha} \mathrm{P}(d | d', \alpha) \qquad \forall v \in \mathcal{V}, d \in \mathrm{D}_v \cup \{g\} \tag{C9}$$

$$out^v(d) = \sum_{\alpha \in \mathrm{A} \cup \{a_g\}} x^v_{d,\alpha} \qquad \forall d \in \mathrm{D}_v \tag{C10}$$

$$out^v(d) - in^v(d) = p^v_0(d) \qquad \forall v \in \mathcal{V}, d \in \mathrm{D}_v \tag{C11}$$

$$in^v(g) = 1 \qquad \forall v \in \mathcal{V} \tag{C12}$$

$$\sum_{d_i \in \mathrm{D}_{v_i}} x^{v_i}_{d,\alpha} = \sum_{d_j \in \mathrm{D}_{v_j}} x^{v_j}_{d,\alpha} \qquad \forall v_i, v_j \in \mathcal{V}, \alpha \in \mathrm{A} \tag{C13}$$

$$\sum_{\langle \boldsymbol{m},s\rangle \in \hat{\mathrm{S}} \cap Accept(i)} in(\boldsymbol{m}, s) \in z_i \qquad \forall i \in \{1, \ldots, k\} \tag{C14}$$

Projection Occupation Measure Heuristic (C9–C12). For each SAS$^+$ variable $v \in \mathcal{V}$, these constraints represent the projection of $\mathcal{T}$ onto v. This projection is an SSP in itself over the state space $\mathrm{D}_v \cup \{g\}$ where g represents the sink of the overall problem (see (Trevizan, Thiébaux, and Haslum 2017) for formal definition). The variables $x^v_{d,\alpha}$ are the occupation measures for the projection onto v. The projections are used for obtaining a lower bound on the expected cost of reaching the goal set $\mathrm{G}^\times$ of the overall problem from Γ.

Tying constraints (C13). These constraints tie the projections onto the different $v \in \mathcal{V}$ together. In order to avoid the complexity of the original problem, the projections are tied using the following relaxation: for all $\alpha \in \mathrm{A}$, the expected number of times action a is applied over the entire projection must be the same for every projection.

PLTL Constraint (C14). These linear constraints enforce the PLTL constraints. The left-hand side of the constraints follows from the definition of $C_i^\times$ and $T_i^\times$. Due to the lack of domain-independent lower bounds for PLTL constraints, i^2-dual can only handle intervals z_i that are left-closed and starting at 0, thus $\mathbf{P}_{\in z_i}\,\psi_i$ is represented using two constraints: $\mathbf{P}_{\in [0,\overline{z}_i]}\,\psi_i$ and $\mathbf{P}_{\in [0,1-\underline{z}_i]}\,\neg\psi_i$.

The objective function of LP1 minimises the expected primary cost of reaching the fringe Γ and the goal states in $\hat{S}$ (first and second summation, respectively) plus the heuristic estimate to solve the problem from the states reached in Γ to the goal set $G^\times$ of the original problem (third summation). Due to the tying constraints (C13), any variable $v' \in \mathcal{V}$ can be used in the third summation. Notice the updates in the search space explored so far and the heuristics estimates happens together and in the same LP, thus the search and heuristic computation work in unison instead of the search method driving the heuristic computation.

Once LP1 is solved, the fringe states reachable by its optimal solution are expanded and a new iteration of i^2-dual is performed. i^2-dual stops when all the flow injected in the initial state reaches the goal set $G^\times$. The optimal policy $\pi^*_{C^\times}$ is encoded in the optimal solution x^* of LP1 in the last iteration of i^2-dual: $\pi^*_{C^\times}(\langle \boldsymbol{m}, s\rangle, \alpha) = \frac{x^*_{\boldsymbol{m},s,\alpha}}{\text{out}(\boldsymbol{m},s)}$ for all $\langle \boldsymbol{m}, s\rangle \in S^\times$ and $\alpha \in A(s)$ such that $\text{out}(\boldsymbol{m}, s) > 0$.

Heuristic for PLTL Constraints

Since the secondary action costs $C_i^\times$ of $C^\times$ are always zero, the heuristics embedded in i^2-dual is unable to estimate the probability of ψ_i being true; therefore, i^2-dual performs heuristic search only for the primary cost. As we show in our experiments, this approach is not enough to solve large MO-PLTL SSPs. We address this issue by introducing a heuristic for the PLTL constraints, i.e., a function that can prioritize the search on $S^\times$ according to a given PLTL constraint ψ_i.

Formally, a *heuristic for* ψ_i is any function $h^{\psi_i} : S^\times \to \mathbb{R}_+$ that estimates the probability of ψ_i being satisfied. h^{ψ_i} is an *admissible heuristic* if, for all $s^\times \in S^\times$, $h^{\psi_i}(s^\times) \geq \max_{\pi\in\Pi^*} V_i^\pi(s^\times)$ where Π^* is the set of optimal policies for $C^\times$ and V_i^π is the policy π value function for the secondary cost associated with ψ_i (Eq. 1). Thus, the trivial admissible heuristic for ψ_i is the always-1 function. In this paper, we consider only admissible heuristics because they do not prune feasible solutions from the search, thus, the soundness, completeness and optimally guarantees of i^2-dual are preserved.

In order to be able to integrate our novel heuristic to i^2-dual and take advantage of the unison search property of i^2-dual, we propose a heuristic based on projections onto the Nondeterministic Büchi Automaton (NBA) of each LTL formula. To get a good estimate from an NBA projection, we need to handle both the non-determinism of the NBA transitions as well as the probabilistic effects of the SAS$^+$ actions. We accomplish this by modelling the NBA projection as a relaxed SSP over the NBA states.

Formally, let $\mathcal{B}_i = (Q_i, S, \Delta_i, \chi_{\text{init},i}, F_i)$ be the NBA for the PLTL constraint ψ_i. Given an effect e of a probabilistic SAS$^+$ action $\alpha \in A$ and $s \in S$, *e is consistent with s* if $res(pre(\alpha), e) \subseteq s$. We call $\text{Comp}(e, q)$ the set of successor states of $q \in Q_i$ compatible with effect e of action α. Formally $\text{Comp}(e, q) = \{q' \in Q_i | \exists (q, s, q') \in \Delta_i, e \text{ is consistent with } s\}$.

The relaxed SSP representing the projection of $C^\times$ onto $\mathcal{B}_i$ is $\mathcal{S}^{\psi_i} = (Q_i, \chi_{\text{start}}, F_i, B^{\psi_i}, P^{\psi_i})$ whose set of states Q_i is that of $\mathcal{B}_i$ and whose set of goal states F_i is the set of accepting states of $\mathcal{B}_i$. This relaxed SSP has a **non-deterministic initial state** $\chi_{\text{start}} \subseteq Q_i$, i.e., no probability distribution over χ_{start} is given. B^{ψ_i} is the set of actions and $B^{\psi_i}(q) = \{\beta_{\alpha,p} | p \in \bigtimes_{e\in eff(\alpha)} \text{Comp}(e, q)\}$ is the set of actions applicable in any $q \in Q_i$. Lastly, $P^{\psi_i}(q'|q, \beta_{\alpha,p}) = \sum_{e\in eff(\alpha), p[e]=q'} \Pr_\alpha(e)$, is the probability of transitioning from q to q' after applying $\beta_{\alpha,p} \in B^{\psi_i}(q)$, where $p[e]$ is the element in the e-th coordinate of p. $\mathcal{S}^{\psi_i}$ does not have a cost function and we are interested in finding a solution to $\mathcal{S}^{\psi_i}$ that maximizes the probability of reaching the goal set, that is, the accepting states of $\mathcal{B}_i$.

Our heuristic for a PLTL constraint, called the NBA projection heuristic and denoted by $h^{\psi_i}_{\text{BA}}$ for the i-th PLTL constraint, is formally presented in LP2. $h^{\psi_i}_{\text{BA}}$ takes as input a non-empty subset χ_{start} of Q_i and returns an upper bound on the probability of ψ_i being true when χ_{start} is used as the non-deterministic initial state of $\mathcal{B}_i$.

$$\min_{x^{\psi_i}, p^{\psi_i}} \quad \sum_{q\in Q_i\setminus F_i} x^{\psi_i}_{q,\alpha_{\text{sink}}} \tag{LP2}$$

$$\text{s.t.} \quad x^{\psi_i}_{q,\beta} \geq 0 \qquad \forall q \in Q_i, \beta \in B^{\psi_i}(q) \cup \{\alpha_{\text{sink}}\} \tag{C15}$$

$$p^{\psi_i}_q \geq 0 \qquad \forall q \in \chi_{\text{start}} \tag{C16}$$

$$x^{\psi_i}_{q,\alpha_{\text{sink}}} + \sum_{\beta\in B^{\psi_i}(q)} x^{\psi_i}_{q,\beta} - \sum_{\substack{q'\in Q_i \\ \beta\in B^{\psi_i}(q')}} P(q|q',\beta) x^{\psi_i}_{q',\beta} = p^{\psi_i}_q \qquad \forall q \in \chi_{\text{start}} \tag{C17}$$

$$x^{\psi_i}_{q,\alpha_{\text{sink}}} + \sum_{\beta\in B^{\psi_i}(q)} x^{\psi_i}_{q,\beta} - \sum_{\substack{q'\in Q_i \\ \beta\in B^{\psi_i}(q')}} P(q|q',\beta) x^{\psi_i}_{q',\beta} = 0 \qquad \forall q \in Q_i \setminus \chi_{\text{start}} \tag{C18}$$

$$\sum_{q\in\chi_{\text{start}}} p^{\psi_i}_q = \sum_{q\in Q_i} x^{\psi_i}_{q,\alpha_{\text{sink}}} = 1 \tag{C19}$$

The variables of LP2 are: the occupation measures $x^{\psi_i}_{q,\alpha}$ for this projection; and the input flows $p^{\psi_i}_q$ representing how the flow injected into the network is distributed within the states $q \in \chi_{\text{start}}$ of the non-deterministic initial state. While $x^{\psi_i}_{q,\alpha}$ is analogous to $x^v_{d,\alpha}$ of LP1, $p^{\psi_i}_q$ have no counterpart since they encode the non-deterministic initial state of $\mathcal{B}_i$.

Notice that an artificial action α_{sink} is used in LP2 (C15, C17, C18 and C19). This action represents the deterministic transition from a state $q \in Q_i$ to the sink and it is applicable in all states (C17 and C18). The source and sink constraints C19 enforces that 1 unit enters and leaves the network. C18 is the set of flow preservation constraints that, for all states $q \notin \chi_{\text{start}}$, forces the flow leaving q to the sink and other states $q' \in Q_i$ to equal the flow entering q. Similarly, C17 is the flow preservation constraint for source states.

The objective function of LP2 minimises the flow going into the sink from the non-accepting states of $\mathcal{B}_i$, thus it

maximises the probability of reaching an accepting state $q \in F_i$ from χ_{start}. The admissibility of $h^{\psi_i}_{BA}$ is formalized by Theorem 7 and its proof is based on the fact that the trivial admissible heuristic for the PLTL constraints is the always-1 function and that the LP can move the flow from any state $q \in Q_i$ to a state $q' \in F_i$ except when q is part of a non-accepting bottom end component of $\mathcal{B}_i$. We refer to the Appendix for the complete proof.

Theorem 7 *$h^{BA}_{\psi_i}$ is an admissible heuristic for ψ_i.*

As the insights for the proof for Theorem 7 suggest, LP2 is encoding the problem of avoiding the non-accepting bottom end components of $\mathcal{B}_i$. Although there are more efficient ways of solving this problem (e.g., a look-up table), our encoding as a projection of the MO-PLTL SSP over ψ_i pays off by being able to be integrated to i^2-dual. The integration is done by tying the actions in the projection onto each ψ_i to the projections onto the state variables $v \in \mathcal{V}$ of the probabilistic SAS$^+$ task. By tying all projections together, both SAS$^+$ and ψ_i projections, we force them to reach a relaxed agreement over their solutions for reaching the original set of goals $G^\times$ from the current fringe states Γ. This agreement provides better heuristic estimates for both the primary cost function and PLTL constraints.

PLTL-dual

Our last contribution is the integration of $h^{\psi_i}_{BA}$ to i^2-dual. The obtained algorithm, PLTL-dual, consists of the same iterative procedure as i^2-dual but LP1 is replaced by LP3. We assume that the mode space is the NBA mode. A visual representation of LP3 is presented in Figure 1 and its constraints can be categorized as:

i^2-dual constraints (C3 – C13). All constraints except the PLTL constraint (C14)

Non-deterministic Multiplexer (C21 – C23). The non-deterministic counterpart of the state variable multiplexer (C7 – C8). This multiplexer represents the non-deterministic distribution of flow from the mode m_i to the states $q \in m_i$. $\Gamma^{\psi_i} = \{m_i \in M_i | \exists \langle \boldsymbol{n}, s\rangle \in \Gamma \text{ s.t. } n_i = m_i\}$ represents all the observed NBA modes m_i of ψ_i in the fringe Γ and $\mathcal{D}(m_i, \Gamma^{\psi_i})$ denotes the NBA states $q \in m_i$ not present in any other mode $n_i \in \Gamma^{\psi_i}$, formally, $\mathcal{D}(m_i, \Gamma^{\psi_i}) = m_i \setminus (\cup_{n_i \in \Gamma^{\psi_i} | n_i \neq m_i} n_i)$.

NBA Projection Heuristic (C17–C18). The flow preservation constraints for $h^{\psi_i}_{BA}$ for each ψ_i.

PLTL sink constraint (C24). The replacement of the sink constraint of $h^{\psi_i}_{BA}$. The new sink extracts the same amount of flow that the non-deterministic multiplexer injected in each NBA projection.

NBA Tying constraints (C25). These constraints tie the NBA projections to the state variable projections. Any state variable $v \in \mathcal{V}$ can be used in these constraints since all state variables are already tied together by C13.

PLTL constraints (C26). These constraints replace C14 and contain the heuristic estimation provided by $h^{\psi_i}_{BA}$.

$$\min_{x} \sum_{\langle \boldsymbol{m}, s\rangle \in \hat{S}, \alpha \in A(s)} x_{\boldsymbol{m},s,\alpha} C_0^\times(\alpha) + \sum_{\langle \boldsymbol{m}, s\rangle \in \hat{S} \cap G^\times} \text{in}(\boldsymbol{m}, s) T_0^\times(\langle \boldsymbol{m}, s\rangle) + \sum_{d \in D_{v'}, \alpha \in A} x^{v'}_{d,\alpha} C_0^\times(\alpha) \quad \text{(LP3)}$$

$$\text{s.t.} \quad x_{s,\alpha} \geq 0, x^v_{d,\alpha} \geq 0, x^{\psi_i}_{q,\alpha} \geq 0, p^{\psi_i}_q \geq 0 \quad \text{(C20)}$$

constraints C3 – C13

$$\sum_{\langle \boldsymbol{m}, s\rangle \in \Gamma | m_i = \{q\}} \text{in}(\boldsymbol{m}, s) \leq p^{\psi_i}_q \leq \sum_{\langle \boldsymbol{m}, s\rangle \in \Gamma | q \in m_i} \text{in}(\boldsymbol{m}, s) \qquad \forall i \in \{1, \ldots, k\}, q \in Q_i \quad \text{(C21)}$$

$$\sum_{q \in \mathcal{D}(m, \Gamma^{\psi_i})} p^{\psi_i}_q \leq \sum_{\langle \boldsymbol{n}, s\rangle \in \Gamma | n_i = m} \text{in}(\boldsymbol{n}, s) \leq \sum_{q \in m} p^{\psi_i}_q \qquad \forall i \in \{1, \ldots, k\}, m \in \Gamma^{\psi_i} \quad \text{(C22)}$$

$$\sum_{q \in Q_i} p^{\psi_i}_q = \sum_{\langle \boldsymbol{m}, s\rangle \in \Gamma} \text{in}(\boldsymbol{m}, s) \qquad \forall i \in \{1, \ldots, k\} \quad \text{(C23)}$$

$$\sum_{q \in Q_i} x^{\psi_i}_{q,\alpha_{\text{sink}}} = \sum_{\langle \boldsymbol{m}, s\rangle \in \Gamma} \text{in}(\boldsymbol{m}, s) \qquad \forall i \in \{1, \ldots, k\} \quad \text{(C24)}$$

constraints C17 – C18 $\qquad \forall i \in \{1, \ldots, k\}$

$$\sum_{d \in D_v} x^v_{d,\alpha} = \sum_{q \in Q_i} x^{\psi_i}_{q,\alpha} \qquad \forall i \in \{1, \ldots, k\}, \alpha \in A \quad \text{(C25)}$$

$$\sum_{\langle \boldsymbol{m}, s\rangle \in \hat{S} \cap \text{Accept}(i)} in(\boldsymbol{m}, s) + \sum_{q \in F_i} x^{\psi_i}_{q,\alpha_{\text{sink}}} \in z_i \qquad \forall i \in \{1, \ldots, k\} \quad \text{(C26)}$$

LP3 assumes that the mode for each PLTL constraint ψ_i is the NBA $\mathcal{B}_i$ associated with ψ_i. This presents two key computational advantages: (i) the NBA projection heuristics do not need to compute $\mathcal{B}_i$ since they were already computed to be used as modes; and (ii) the non-deterministic multiplexer can easily relate the mode $m_i \subseteq Q_i$ and the NBA projection states $q \in Q_i$ since they are defined over the same set Q_i of $\mathcal{B}_i$. Nonetheless, PLTL-dual can be used with any mode as long as a translation from the non-NBA mode m_i to a subset of Q_i is provided for each PLTL constraint.

Experiments

In this section we empirically evaluate our heuristic search algorithms and compare their performance with that of Prism, a state-of-the-art model-checker implementing the static approach (Kwiatkowska, Norman, and Parker 2011). We enforce a 30-minutes and 4-Gb cut-off for all experiments, and report results averaging 30 runs of each algorithm for each problem taken from the following two domains.

Factory. This domain features an assembly line with n machines, where each machine m_i produces a part p_i starting from the part p_{i-1} produced by machine m_{i-1}. There are actions for turning a machine on or off (cost 1), and for producing p_i using machine m_i once m_i is on and p_{i-1} is available. This makes p_{i-1} unavailable. The k machines $m_2 \ldots m_{k+1}$ are unreliable and fail to produce the new part p_i 20% of the time, but still make p_{i-1} unavailable. Production cost for reliable (resp. unreliable) machines is 5 (resp. 3). Initially, all machines are off and p_0 is available. In the goal, the machines are off again and part p_n has been produced. The two MO-PLTL constraints ($\mathbf{P} = 1$) are that (1) m_1 is eventually started, and (2) once m_{i-1} stops for good, m_i has to be on, and then the same hand-shake applies to the next machine down the line and so on: $\mathbf{G}\,(on(m_1) \Rightarrow (on(m_1)\,\mathbf{U}\,(on(m_2) \wedge \mathbf{G}\,\neg on(m1) \wedge (on(m_2)\,\mathbf{U}\,(on(m_3) \wedge \mathbf{G}\,\neg on(m_2) \wedge \ldots \wedge (on(m_{n-1}) \wedge \mathbf{U}\,(on(m_n) \wedge \mathbf{G}\,\neg on(m_n)))\ldots)))))$.

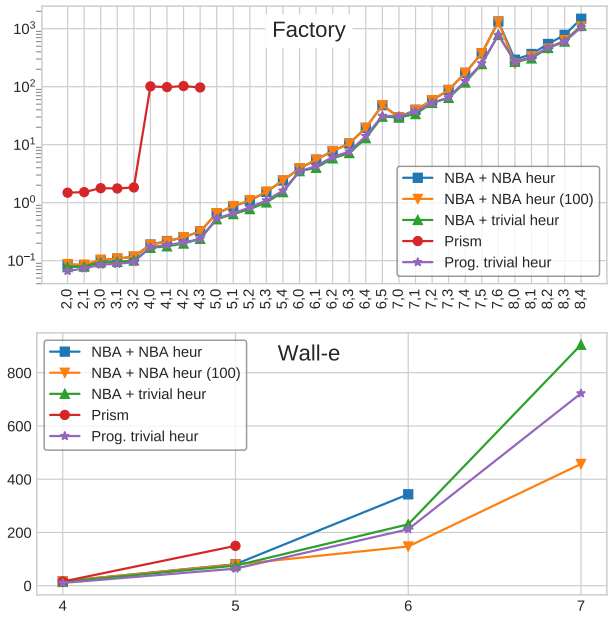

Figure 2: Time in seconds to solve: factory problems n, k ($n \in 2..8$, $k \in 0..(n-1)$); and Wall-e problem for $n \in 4..7$.

Wall-e. In this domain, Wall-e and Eve are in a corridor with n distinct locations $l_1 \ldots l_n$ and n rooms $r_1 \ldots r_n$. Location l_i is connected to room r_i and to locations l_{i-1} and l_{i+1}. Wall-e and Eve can be in any of these locations and rooms, either separately or together. They can move to a connected location, enter or exit a connected room, either separately or together. All actions are deterministic and have cost 1, except exiting a room together which has cost 5 and fails with 10% probability. Wall-e starts in l_1 and Eve in r_2, and the goal has Wall-e in l_n. The MO-PLTL constraints specify that: (1) they must eventually be together ($\mathbf{P} \geq 0.5$); (2) once they're together they remain together ($\mathbf{P} = 1$); (3) Eve must be at most 3 steps away from a room until they are together ($\mathbf{P} \geq 0.8$); (4) Eve visits the first 3 rooms r_1, r_2, r_3 ($\mathbf{P} = 1$); and (5) Wall-e never visits any room twice, except possibly r_n ($\mathbf{P} \geq 0.8$).

Algorithms. The algorithms considered are: (a) PLTL-dual, that is, i^2-dual with the NBA mode and NBA projection heuristic; (b) PLTL-dual(100), which is like PLTL-dual except that the NBA projection heuristic is only used for NBAs with less than 100 states – for the other formulas, the trivial heuristic is used; (c) i^2-dual with the NBA mode and the trivial heuristic for all formulas; (d) i^2-dual with the progression mode and the trivial heuristic for all formulas; and (e) the multi-objective version of Prism with the -lp option.[4] In our implementation, we use ltl3ba to produce the NBAs.

Results. The graphs in Figure 2 show the time spent by each algorithm on problems from the Factory domain (log scale) with $n = 2..8$ machines including $k = 0..(n-1)$ unreliable machines, and from the Wall-e domain with $n = 4..7$ rooms.

[4]Without this option, Prism produces an error on both domains.

We find it convenient to analyse the results by means of answering the following questions:

What is the best mode? To answer this question, we need to compare both modes using the same heuristics, i.e., the trivial heuristics. In this case, the progression mode outperforms the NBA mode in the Wall-e domain (25% faster for $n = 7$) while being statistically tied in the Factory domain. The reason for this performance advantage is that progression uses the observed information so far to simplify its search space thus reducing the overall number of states expanded, e.g., 8% less expanded states in Wall-e #7.

Is the NBA projection heuristic effective? Yes when the formulas are not trivial to check and the NBAs are not too large. The first condition is usual for heuristic search algorithms because, in easy search problems, a less accurate but cheap to compute heuristic is good enough. Constraint (2) of the Factory domain is an example of a constraint trivial to check since turning on a machine in the wrong order violates it. The second condition is an issue because large NBAs result in projections with a large number of LP variables. For instance, the NBA for constraint (5) of Wall-e #6 has 112 states and its projection uses 19863 LP variables. This represents about 40% of the total LP variables in the last LP solved by PLTL-dual in Wall-e #6. An example of good guidance provided by the NBA projection heuristic is constraint (3) of the Wall-e domain. This constraint has a small NBA (11 states) and violations can be detected early by using a look-ahead of size 3. Since the NBA projection heuristic performs search in the mode space, it is able to do early pruning for this constraint while the trivial heuristic, which cannot look beyond the current state, is unable to do.

What is the best algorithm? Although progression is the best mode when using the trivial heuristic, the best algorithm is PLTL-dual(100): it is statistically tied with i^2-dual with progression for the large Factory problems ($n = 8$) while it dominates all other algorithms in the Wall-e domain. As expected, the static approach employed by Prism is outperformed by our heuristic search approaches due to the prohibitive size of the DRA required by it. For instance, the DRA for constraint (2) of Factory #4,3 and constraint (5) of Wall-e #5 has, respectively, 29979 and 2857 states while their NBA has only 13 and 48 states, respectively.

Conclusion and Future Work

We have provided the first heuristic search algorithm for SSPs with MO-PLTL constraints, and demonstrated both practical and in certain cases worst-case complexity improvements over state-of-the-art algorithms. Our approach performs an on-the-fly translation of the MO-PLTL SSP into a C-SSP, which is solved by extending recent heuristic search approaches for C-SSPs and guiding them using novel linear programming heuristics for MO-PLTL constraints. In the future, we plan to improve our NBA heuristics, design heuristics that work together with progression, and extend the scope of heuristic search to SSPs with larger subsets of PCTL*.

Acknowledgement

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

References

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

Bacchus, F., and Kabanza, F. 1998. Planning for Temporally Extended Goals. *Ann. Math. Artif. Intell.* 22(1-2):5–27.

Backström, C. 1992. Equivalence and Tractability Results for SAS$^+$ Planning. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning*.

Baier, C., and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Baier, J. A., and McIlraith, S. A. 2006. Planning with First-Order Temporally Extended Goals using Heuristic Search. In *Proc. AAAI Conf. on Artificial Intelligence*.

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to Act Using Real-Time Dynamic Programming. *Artif. Intell.* 72(1-2):81–138.

Bauer, A., and Haslum, P. 2010. LTL Goal Specifications Revisited. In *Proc. Int. Conf. on Artificial Intelligence*.

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 Geffner, H. 2005. mGPT: A Probabilistic Planner Based on Heuristic Search. *J. Artif. Intell. Res.* 24:933–944.

Bonet, B., and Geffner, H. 2012. Action Selection for MDPs: Anytime AO* Versus UCT. In *Proc. AAAI Conf. on Artificial Intelligence*.

Camacho, A.; Triantafillou, E.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2017. Non-Deterministic Planning with Temporally Extended Goals: LTL over Finite and Infinite Traces. In *Proc. AAAI Conf. on Artificial Intelligence*.

Courcoubetis, C., and Yannakakis, M. 1995. The Complexity of Probabilistic Verification. *J. ACM* 42(4):857–907.

De Giacomo, G., and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Proc. Int. Joint Conf. on Artificial Intelligence*.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on Finite Traces. In *Proc. Int. Joint Conf. on Artificial Intelligence*.

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

Ding, X. C.; Smith, S.; Belta, C.; and Rus, D. 2014. Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints. *IEEE Trans. Automat. Contr.* 59(5):1244–1257.

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*.

Edelkamp, S. 2006. On the Compilation of Plan Constraints and Preferences. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Etessami, K.; Kwiatkowska, M. Z.; Vardi, M. Y.; and Yannakakis, M. 2008. Multi-Objective Model Checking of Markov Decision Processes. *Logical Methods in Computer Science* 4(4).

Forejt, V.; Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2011. Automated Verification Techniques for Probabilistic Systems. In *Proc. Int. School on Formal Methods for the Design of Computer*.

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

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Kwiatkowska, M. Z., and Parker, D. 2013. Automated Verification and Strategy Synthesis for Probabilistic Systems. In *Proc. Int. Symp. on Automated Technology for Verification and Analysis*.

Kwiatkowska, M. Z.; Norman, G.; and Parker, D. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Proc. Int. Conf. on Computer Aided Verification*.

Lacerda, B.; Parker, D.; and Hawes, N. 2015. Optimal Policy Generation for Partially Satisfiable Co-Safe LTL Specifications. In *Proc. Int. Joint Conf. on Artificial Intelligence*.

Lacerda, B.; Parker, D.; and Hawes, N. 2017. Multi-Objective Policy Generation for Mobile Robots under Probabilistic Time-Bounded Guarantees. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Sprauel, J.; Kolobov, A.; and Teichteil-Königsbuch, F. 2014. Saturated Path-Constrained MDP: Planning under Uncertainty and Deterministic Model-Checking Constraints. In *Proc. AAAI Conf. on Artificial Intelligence*.

Teichteil-Königsbuch, F.; Vidal, V.; and Infantes, G. 2011. Extending Classical Planning Heuristics to Probabilistic Planning with Dead-Ends. In *Proc. AAAI Conf. on Artificial Intelligence*.

Thiébaux, S.; Gretton, C.; Slaney, J. K.; Price, D.; and Kabanza, F. 2006. Decision-Theoretic Planning with non-Markovian Rewards. *J. Artif. Intell. Res.* 25:17–74.

Torres, J., and Baier, J. A. 2015. Polynomial-Time Reformulations of LTL Temporally Extended Goals into Final-State Goals. In *Proc. Int. Joint 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*.

Trevizan, F. W.; Teichteil-Königsbuch, F.; and Thiébaux, S. 2017. Efficient solutions for Stochastic Shortest Path Problems with Dead Ends. In *Proc. Conf. on Uncertainty in Artificial Intelligence*.

Trevizan, F.; Thiébaux, S.; and Haslum, P. 2017. Occupation Measure Heuristics for Probabilistic Planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*.

Tseitin, G. 1968. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic* 8:115–125.