Progression Heuristics for Planning with Probabilistic LTL Constraints

Ian Mallett, Sylvie Thiébaux, Felipe Trevizan
Research School of Computer Science, The Australian National University
Ian.Mallett@anu.edu.au, Sylvie.Thiebaux@anu.edu.au, Felipe.Trevizan@anu.edu.au

Abstract

Probabilistic planning subject to multi-objective probabilistic temporal logic (PLTL) constraints models the problem of computing safe and robust behaviours for agents in stochastic environments. We present novel admissible heuristics to guide the search for cost-optimal policies for these problems. These heuristics project and decompose LTL formulae obtained by progression to estimate the probability that an extension of a partial policy satisfies the constraints. Their computation with linear programming is integrated with the recent PLTL-dual heuristic search algorithm, enabling more aggressive pruning of regions violating the constraints. Our experiments show that they further widen the scalability gap between heuristic search and verification approaches to these planning problems.

1 Introduction

In safety-critical planning applications, optimising performance is not enough, and utility must be traded-off against the risk of jeopardizing complex mission goals and constraints. For instance, consider a Mars rover that must gather scientific information and send it to earth (Zilberstein et al. 2001). Safety requirements and constraints include avoiding collision against obstacles, not traversing into unsafe terrains, maintaining safe operational temperature and battery levels, and so on. The rover should seek to optimise expected science returns while remaining safe by proactively keeping the probability of violating each of the above constraints within acceptable levels.

Such planning problems can be modelled as stochastic shortest path problems (SSP) subject to multi-objective probabilistic linear temporal logic (MO-PLTL) constraints $\mathsf{Pr}(\psi_i){\in}z_i$, where the ψ_i are LTL formulae and $z_i \subseteq [0,1]$ are intervals bounding their respective probabilities (Baumgartner, Thiébaux, and Trevizan 2018). Optimal solutions to MO-PLTL SSPs take the form of stochastic finite-memory policies, where the probability of the next action to perform depends both on the current state of the environment and on a mode used to track the truth value of the LTL formulae.

Variants of MO-PLTL SSPs have been extensively studied by the automated verification community (Baier and Katoen 2008), and their resolution is supported by tools such as the PRISM model-checker (Kwiatkowska, Norman, and Parker 2011). However, these approaches build, upfront, the entire state space of the MO-PLTL SSP, i.e. the synchronous product of the modes and environment states (Etessami et al. 2008; Forejt et al. 2011; Kwiatkowska and Parker 2013). The prohibitive size of this construction, polynomial in the huge number of reachable environment states and in the worst case double exponential in the size of the formulae, precludes applicability to the factored state spaces found in AI planning.

Recently, heuristic search has become the state of the art approach for solving planning problems modelled as factored MO-PLTL SSPs. In particular, Baumgartner et al. (2018) introduced PLTL-dual, an algorithm which builds the state space of the MO-PLTL SSPs on-the-fly from the factored representation. PLTL-dual applies linear programming to increasingly larger subsets of the reachable state space, guided by admissible heuristics to prune regions that cannot satisfy the constraints or are too costly to form part of an optimal policy. When guided by informative heuristics, PLTL-dual only needs to expand a fraction of the reachable state space to find an optimal policy satisfying the constraints. This yields significant scalability improvements over the conventional approach implemented in PRISM.

Unfortunately, effective heuristic guidance can be difficult to obtain. Classical planning enjoys a multitude of admissible heuristics (Bonet and Geffner 2001; Helmert, Haslum, and Hoffmann 2007; Helmert and Domshlak 2009; Pommerening et al. 2014). However, it is only recently that the first SSP heuristics taking into account both probabilities and costs have been proposed (Trevizan, Thiébaux, and Haslum 2017). Moreover, only a couple of heuristics exist for deterministic planning with LTL-like constraints (Baier, Bacchus, and McIlraith 2009; Bienvenu, Fritz, and McIlraith 2011), and heuristic search for SSPs with probabilistic LTL constraints is in its infancy. Baumgartner et al. started to investigate the latter, and devised projection heuristics based on a representation of LTL formulae and policy modes in terms of non-deterministic Büchi automata (NBA) (Vardi and Wolper 1994; Babiak et al. 2012). Their experimental results show that the size of the linear programs (LPs) required to represent the projections of NBAs grows unacceptably large, so that the projection heuristic with NBAs larger than 100 states obtains worse results than the trivial heuristic ($\mathsf{Pr}(\psi_i) = 1\ \forall i$). To avoid generating NBAs, they also experimented with modes obtained by formula progression

(Bacchus and Kabanza 1998) and the trivial heuristic, but were unable to devise heuristics based on progression.

This paper presents the first admissible heuristics based on progression for probabilistic LTL. These heuristics overestimate the probability of an LTL formula being satisfiable by completions of a partial policy. We present two heuristic estimates. The first loosely ties together various projections of the MO-PLTL SSP over subsets of state variables chosen via a principled examination of the formulae. The second applies a further relaxation that decomposes the formulae, sums the probabilities of disjuncts, and averages the probabilities of conjuncts. We show how to embed the computation of these heuristics into the LPs used by PLTL-dual, which makes optimisation and heuristic estimation synergic and avoids repeated calls to heuristic estimators. Our results show that the progression heuristics are competitive and further increase the superiority of PLTL-dual over PRISM.

The paper starts with background on MO-PLTL SSPs in Section 2. Sections 3 and 4 describe the projection and decomposition heuristics, respectively, and Section 5 their integration into PLTL-dual. Section 6 gives experimental results and Section 7 concludes with related and future work.

2 Background

2.1 MO-PLTL SSPs

A *Stochastic Shortest Path problem* (SSP) with *Multi-Objective Probabilistic Temporal Logic* (MO-PLTL) constraints is a tuple $\langle L, \mathsf{S}, s_0, \mathsf{G}, \mathsf{A}, P, C, \vec{\uppsi}\rangle$ where: L is a set of atoms, $\mathsf{S} \subseteq 2^L$ is the finite set of states; $s_0 \in \mathsf{S}$ is the initial state; $\mathsf{G} \subseteq \mathsf{S}$ is the non-empty set of goal states; A is the finite set of actions and we write $\mathsf{A}(s)$ for the set of actions applicable in state s; $P(s'|s,a)$ is the probability of transitioning from s to s' when action $a \in \mathsf{A}(s)$ is applied in s; $C(a) \in \mathbb{R}^*_+$ is the immediate cost of applying action a; and $\vec{\uppsi}$ is a vector of k probabilistic LTL constraints. Each constraint is of the form $\uppsi_i \equiv \mathsf{Pr}(\psi_i) \in z_i$ where ψ_i is a linear temporal logic (LTL) formula over atoms in L, and $z_i \subseteq [0,1]$ is an interval bounding its probability.

We assume that the reader is familiar with *Linear Temporal Logic* (LTL) and refer to (Baier and Katoen 2008) for a detailed account. Briefly, the standard version of LTL specifies properties of infinite sequences of states (or paths). It extends propositional logic with the operators $\mathbf{X}\varphi$, which holds if φ holds at the next position in the sequence, and $\psi\mathbf{U}\varphi$, which specifies that ψ must hold at every point in the sequence until φ holds. We also use the operator $\psi\mathbf{R}\varphi = \neg(\neg\psi\mathbf{U}\neg\varphi)$ which is required by the transformation to negation normal form assumed in Subsection 3.2.

The standard semantics of LTL specifies when an *infinite* path q satisfies the formula φ, which we write $q \models \varphi$. However, in planning, the sequences we seek are finite and end in a goal state.[1] For a *finite* path p ending in state $\mathrm{last}(p)$, we follow Baumgartner et al. (2018) in using the *Infinite Extension Semantics* (Bacchus and Kabanza 1998; Bauer and Haslum 2010), which stipulates that p satisfies φ iff the infinite sequence that loops in the final state of p satisfies φ, i.e. iff $p\,\mathrm{last}(p)^\omega \models \varphi$. Our heuristics can trivially be adapted to slightly different finite path semantics, such as f-FOLTL (Baier and McIlraith 2006) or LTL$_f$ (De Giacomo and Vardi 2013). Note that we do not require the formulas to be either safe or co-safe, as in e.g. (Lacerda, Parker, and Hawes 2015).

A solution to an MO-PLTL SSP is a *stochastic finite-memory policy*, $\pi\colon \mathsf{S} \times \mathcal{M} \times \mathsf{A} \to [0,1]$ where $\mathcal{M}$ is a set of mode vectors $\vec{m}$ which act as memory for the policy, starting from an initial mode vector $\vec{m_0}$. Intuitively, each element m_i of $\vec{m}$ is the memory that keeps track of the satisfaction of the LTL formula ψ_i, and is updated upon transitioning between states. For the purpose of this paper, we assume that this update function is *deterministic*[2] (i.e., a single mode $\vec{m'}$ may result from updating mode $\vec{m}$ when the state changes from s to s'). The policy maps the current state s and mode vector to a probability distribution over the applicable actions $\mathsf{A}(s)$. Due to the determinism of the mode update function, the policy corresponds to a Markov chain and its execution induces a probability distribution over the sequences of states of the MO-PLTL SSP. A valid policy must reach the goal G with probability 1 and satisfy the PLTL constraints, i.e. the probability mass of the sequences satisfying ψ_i must fall within z_i. An optimal policy is a valid policy with minimal expected cost.

We consider policies whose modes are obtained by *progression* of the LTL formulae (Bacchus and Kabanza 1998). As shown by Baumgartner et al., this does not compromise optimality. Formula progression is a technique to track an agent's progress towards satisfying an LTL formula, as the sequence of states followed when executing the policy unfolds. Progression looks at the current state s of the sequence and at the formula φ to satisfy, and returns a new formula $\varphi' = \mathrm{prog}(s,\varphi)$ that must be satisfied by the rest of the sequence: $sp \models \varphi$ iff $p \models \varphi'$. Fitting with infinite extension semantics, we also define $\mathrm{idle}(s,\varphi)$ which returns true if and only if $s^\omega \models \varphi$. Using progression, the policy modes are vectors of LTL formulae $\vec{\varphi} = (\varphi_1,\ldots,\varphi_k)$, and upon transitioning to s, are updated with $\mathrm{prog}(s,\varphi_i)$ for all i.

2.2 Solution Approaches for MO-PLTL SSPs

Existing solution approaches compile the MO-PLTL SSP into the problem of minimising the cost of reaching certain accepting states with the required probabilities. This analysis is performed in an augmented state space which is the synchronised product of the regular state space S and the mode space $\mathcal{M}$. More precisely, the augmented state space is $\mathsf{S}^\times = \mathsf{S} \times \mathcal{M}$; the initial state is $t_0 = \langle s_0, \vec{\varphi_0}\rangle$ where the initial mode $\vec{\varphi_0}$ is such that $\varphi_{0i} = \mathrm{prog}(s_0,\psi_i)$ for all i, the set of goal states is $\mathsf{G}^\times = \mathsf{G} \times \mathcal{M}$; the action applicability function is $\mathsf{A}^\times(\langle s,\varphi\rangle) = \mathsf{A}(s)$, the transition probability distribution is $P^\times(\langle s',\vec{\varphi}'\rangle | \langle s,\vec{\varphi}\rangle, a) = P(s'|s,a)$ if $\varphi'_i = \mathrm{prog}(\varphi_i, s')$ for all i and 0 otherwise; and the accepting states with respect to the i^{th} constraint of the MO-PLTL SSP are $\mathsf{T}_i = \{\langle s,\vec{\varphi}\rangle \in \mathsf{G}^\times | \,\mathrm{idle}(s,\varphi_i)\}$.

In theory, this reachability problem can be solved by a linear program, known as the dual LP, whose variables are the

[1] For SSPs the lengths of the finite paths to the goal are unbounded, which is why SSPs are said to have an *indefinite* horizon.

[2] More permissive options are possible, provided that the synchronised product described in Subsection 2.2 is an SSP.

occupation measures $x_{t,a}$ representing the expected number of times action a will be performed in state $t = \langle s,\vec{\varphi}\rangle \in \mathsf{S}^\times$ when executing the policy – see (Altman 1999; Baier and Katoen 2008; Baumgartner, Thiébaux, and Trevizan 2018):

$$
\begin{aligned}
&\min_{x_{t,a}\geq 0} && \sum_{t\in\mathsf{S}^\times, a\in\mathsf{A}^\times(t)} x_{t,a}C(a) && \text{(LP1)}\\
&\text{s.t.} && in(t) = \sum_{t'\in\mathsf{S}^\times, a\in\mathsf{A}^\times(t')} x_{t',a}P^\times(t|t',a) \quad \forall t \in \mathsf{S}^\times && \text{(C1)}\\
& && out(t) = \textstyle\sum_{a\in\mathsf{A}^\times(t)} x_{t,a} \quad \forall t \in \mathsf{S}^\times \setminus \mathsf{G}^\times && \text{(C2)}\\
& && out(t_0) - in(t_0) = 1 && \text{(C3)}\\
& && out(t) - in(t) = 0 \quad \forall t \in \mathsf{S}^\times \setminus (\mathsf{G}^\times \cup \{t_0\}) && \text{(C4)}\\
& && \textstyle\sum_{t_g\in\mathsf{G}^\times} in(t) = 1 && \text{(C5)}\\
& && \textstyle\sum_{t\in\mathsf{T}_i} in(t) \in z_i \quad \forall \psi_i \in \vec{\psi} && \text{(C6)}
\end{aligned}
$$

LP1 can be viewed as a flow problem where $x_{t,a}$ describes the flow leaving state t via action a and $x_{t,a}P(t'|t,a)$ units of flow reaching each successor state t'. The objective represents the total expected cost to reach the goal (sink) from the initial state (source). The functions $in(t)$ and $out(t)$ in C1-C2 are shorthand for the incoming and outgoing flow for state t. C3-C5 represent, respectively, the source, the flow preservation and the sink of the flow network, and C6 enforces the constraints on the probability of reaching an accepting state. The optimal solution x^* of LP1 can be converted into an optimal policy $\pi^*(t,a) = x^*_{t,a}/\sum_{a'\in\mathsf{A}^\times(t)} x^*_{t,a'}$.

LP1, with its $|\mathsf{S}| \times \prod_i 2^{2^{|\psi_i|}} \times |\mathsf{A}|$ variables, is unacceptably large. Hence, Baumgartner et al. (2018) investigated a heuristic search approach, PLTL-dual, which only applies LP1 to a subset of the augmented state space, constructed on the fly, starting from t_0. At each iteration, the search expands the fringe states reachable under the optimal policy found at the previous iteration and runs the LP. The search stops if the LP finds a proper policy as this policy is guaranteed to be optimal. If the MO-PLTL SSP is unsolvable, an infeasible LP will be produced at some iteration when or before the complete augmented state space is generated.

The search is guided by $k+1$ admissible heuristics to evaluate fringe states: a cost heuristic that under-estimates the expected cost of reaching the goal, guiding the search towards cheap policies; and one heuristic for each PLTL constraint that over-estimates[3] the probability of the respective formula, guiding the search towards valid policies. These heuristics map probability distributions $\mathbf{P}$ over the fringe of $\mathsf{S}^\times$ to $\mathbb{R}_+$. The set F of reachable fringe states is the support of $\mathbf{P}$, i.e., $\mathrm{F} = \{\langle s,\vec{\varphi}\rangle \in \mathsf{S}^\times | \mathbf{P}(s,\vec{\varphi}) > 0\}$. A heuristic h_{ψ_i} for PLTL constraint ψ_i is admissible if, for all $\mathbf{P}$, $h_{\psi_i}(\mathbf{P}) \geq \sum_{\langle s,\vec{\varphi}\rangle\in\mathrm{F}} \mathbf{P}(s,\vec{\varphi}) \times \mathsf{Pr}^*(\varphi_i|s)$, where $\mathsf{Pr}^*(\varphi_i|s)$ is the maximum probability of satisfying φ_i from state $\langle s,\vec{\varphi}\rangle$ over the set of optimal policies.

Baumgartner et al. (2018) uses the h^{pom} heuristic (Trevizan, Thiébaux, and Haslum 2017) to estimate cost, and a heuristic h^{BA} based on the NBA representation of LTL formulae to estimate probabilities. However, these heuristics do not scale to NBAs exceeding 100 states.[4] In the rest of this paper, we show that more powerful heuristics for estimating probabilities can be directly obtained from the progressed formulae labelling the fringe states, and present two such heuristics, h^{pom}_ψ which is based on projection and h^{dec}_ψ which is based on a further decomposition of the formula. These heuristics apply to a single constraint, which we write as $\psi \equiv \mathsf{Pr}(\psi) \in z$ to simplify notation.

3 LTL Projection

In this section we introduce our heuristic for PLTL constraints based on a set of *projections*, i.e., a set of smaller MO-PLTL SSPs obtained by ignoring different subsets of atomic propositions. Each of these projections simplify the underlying SSP and one of our novel contributions is to show how to simplify the LTL formulae to obtain a meaningful heuristic. We first describe the *multi-variable projections*, then we introduce our LTL formula projection and show how to integrate different projections into a single heuristic.

3.1 Underlying SSP Projection

For the remainder of this paper, we assume that the set of atomic propositions L is compactly represented as in SAS$^+$ (Bäckström 1992), using a set of multi-valued variables $v \in \mathcal{V}$, each with a finite domain D_v. This allows us to represent each state $s \in \mathsf{S}$ as a set of values, one per state variable, and we denote by $s[v] \in \mathsf{D}_v$ the value of v in s. We also assume that the actions in A are represented compactly using partial valuations over $\mathcal{V}$, i.e., using functions from $\mathcal{V}$ to $\times_{v\in\mathcal{V}}(\mathsf{D}_v \cup \{\bot\})$ where $s[v] = \bot$ denotes that v has no assigned value. Using this representation, an action a consists of: a partial valuation $pre(a)$ denoting its precondition; a set $\mathit{eff}(a)$ of partial valuations representing the effects of the action; and a probability distribution $P_a(\cdot)$ over effects $e \in \mathit{eff}(a)$ representing the probability of e being selected when a is applied. The result of applying an effect e in a state $s \in \mathsf{S}$ is the state $res(s,e) \in \mathsf{S}$ such that, for all state variables v of s, $res(s,e)[v] = e[v]$ if $e[v] \neq \bot$ and $res(s,e)[v] = s[v]$ otherwise.

Given a non-empty set of variables $\mathcal{U} \subset \mathcal{V}$, the projection of a state s onto $\mathcal{U}$ is a state $s^{\mathcal{U}} \in \times_{v\in\mathcal{U}} \mathsf{D}_v$ s.t. $s^{\mathcal{U}}[v] = s[v]$ for all $v \in \mathcal{U}$. The projection of an MO-PLTL SSP onto $\mathcal{U}$ is $\langle L^{\mathcal{U}}, \mathsf{S}^{\mathcal{U}}, s_0^{\mathcal{U}}, \mathsf{G}^{\mathcal{U}}, \mathsf{A}, P^{\mathcal{U}}, C, \psi^{\mathcal{U}}\rangle$ where:

$$
\begin{aligned}
&L^{\mathcal{U}} = \{(v = d) | v \in \mathcal{U}, d \in \mathsf{D}_v\};\\
&\mathsf{S}^{\mathcal{U}} = \bigtimes_{v\in\mathcal{U}} \mathsf{D}_v;\\
&\mathsf{G}^{\mathcal{U}} = \{s^{\mathcal{U}} | s \in \mathsf{G}\};\ \text{and}
\end{aligned}
$$

if a is applicable in $s^{\mathcal{U}} \in \mathsf{S}^{\mathcal{U}}$ (i.e., $pre(a)[v] \in \{\bot, s^{\mathcal{U}}[v]\}$ for all $v \in \mathcal{U}$) then

$$
P^{\mathcal{U}}(s^{\mathcal{U}\prime}|s^{\mathcal{U}}, a) = \sum_{\substack{e\in\mathit{eff}(a)\\ \text{s.t. } s^{\mathcal{U}\prime} = res(s^{\mathcal{U}}, e)}} P_a(e).
$$

Next, we show how to project the PLTL constraint ψ onto $\mathcal{U}$ to obtain $\psi^{\mathcal{U}}$.

[3] Note that over-estimates of probabilities are sufficient: PLTL-dual obviates the need to compute under-estimates by converting every constraint $\mathsf{Pr}(\psi_i) \in [\underline{z_i}, \overline{z_i}]$ into two lower bound constraints: $\mathsf{Pr}(\psi_i) \in [\underline{z_i}, 1]$ and $\mathsf{Pr}(\neg\psi_i) \in [1 - \overline{z_i}, 1]$.

[4] Using the LTL$_f$ semantics (De Giacomo and Vardi 2013) and finite automata would not remedy this issue.

3.2 Formula Projection

When projecting a formula ψ onto a set of variables $\mathcal{U}$, we assume without loss of generality that ψ is in negation normal form without spurious $\top$ and $\bot$, and we relax the PLTL constraint by assuming that each instance of a forgotten variable (in $\mathcal{V} \setminus \mathcal{U}$) is independent of the others. That is, multiple instances of a forgotten variable can variously be assigned true or false at each state in a path. Under these assumptions, all *literals* in ψ containing forgotten variables can be trivially replaced with $\top$, allowing the formula to be further simplified. For instance, a formula $\mathbf{F}(v_1 = 1) \wedge ((v_1 = 3)\mathbf{U}(\neg(v_2 = 1)))$ when projected onto $\{v_2\}$ would become $\top\mathbf{U}(\neg(v_2 = 1))$, preserving the requirement that v_2 must eventually not be 1.

We now explain how to choose suitable sets $\mathcal{U}$ of variables to project onto. The aim of a PLTL heuristic is to extract some information about how to satisfy a constraint; there is no point if after projection the formula is trivial (i.e., it simplifies to $\top$ or $\bot$). Let $\mathrm{minvars}(\psi)$ be the minimal set of variable combinations $\{C_1, \dots, C_n\}$, such that the projection of ψ onto the variables $\mathcal{U}$ is non-trivial if and only if there is some combination $C_j \in \mathrm{minvars}(\psi)$ s.t. $C_j \subseteq \mathcal{U}$. It can clearly be seen that the set $\mathrm{minvars}(\psi)$ can be computed recursively as follows:

$$\begin{aligned}
\mathrm{minvars}(\top) &= \mathrm{minvars}(\bot) = \emptyset \\
\mathrm{minvars}((v = d)) &= \mathrm{minvars}(\neg(v = d)) = \{\{v\}\} \\
\mathrm{minvars}(\psi_1 \wedge \psi_2) &= \mathrm{reduce}(\mathrm{minvars}(\psi_1) \cup \mathrm{minvars}(\psi_2)) \\
\mathrm{minvars}(\psi_1 \vee \psi_2) &= \mathrm{reduce}(\{C_1 \cup C_2 | C_1 \in \mathrm{minvars}(\psi_1), \\
&\qquad\qquad\qquad C_2 \in \mathrm{minvars}(\psi_2)\}) \\
\mathrm{minvars}(\mathbf{X}\psi) &= \mathrm{minvars}(\psi) \\
\mathrm{minvars}(\psi_1 \mathbf{U} \psi_2) &= \mathrm{minvars}(\psi_1 \mathbf{R} \psi_2) = \mathrm{minvars}(\psi_2)
\end{aligned}$$

where the reduce operator removes combinations that are subsumed by others. Consider our earlier example $\psi = \mathbf{F}(v_1 = 1) \wedge ((v_1 = 3)\mathbf{U}(\neg(v_2 = 1)))$. Projecting onto $\{v_1\}$ will preserve the left hand side of the $\wedge$, and $\{v_2\}$ only the right hand side of the $\mathbf{U}$. Note that projecting ψ onto $\{v_1, v_2\}$ is also non-trivial, but any set $\mathcal{U}$ with $\{v_1, v_2\} \subseteq \mathcal{U}$ also has $\{v_1\} \subseteq \mathcal{U}$, so $\{v_1, v_2\}$ does not appear in the *minimal* set of combinations. Hence $\mathrm{minvars}(\psi) = \{\{v_1\}, \{v_2\}\}$. We choose a set of combinations $\mathcal{P}_\psi$ randomly from $\mathrm{minvars}(\psi)$ such that $\mathcal{P}_\psi$ covers the set of variables in ψ. Our heuristic projects the MO-PLTL SSP onto each of these combinations in $\mathcal{P}_\psi$ and ties these projections together as explained below.

3.3 Tying Projections

For each combination C_j in $\mathcal{P}_\psi$, we build a dual LP (LP1) for the projection of the MO-PLTL problem onto C_j. We denote the set of reachable augmented states in this projection $\mathsf{S}^\times_j$.

These projections are tied together by extra tying constraints which enforce that each projection should use each action an equal number of times in expectation:

$$\textstyle\sum_{t \in \mathsf{S}^\times_j} x_{t,a} = \sum_{t \in \mathsf{S}^\times_{j'}} x_{t,a} \qquad \forall C_j, C_{j'} \in \mathcal{P}_\psi, a \in \mathsf{A} \qquad \text{(C7)}$$

The heuristic $h^{\mathrm{pom}}_\psi(\mathbf{P})$ injects $\mathbf{P}(s, \varphi)$ flow into each network at the projection of each fringe state $\langle s, \varphi \rangle \in \mathrm{F}$, and maximises the amount of flow reaching the accepting states of the projection.[5] The heuristic estimate is that maximum.

h^{pom}_ψ is admissible. The proof follows from the fact that the literal independence assumption makes formulae easier to satisfy. Hence any path from a state in F which reaches a goal state and satisfies ψ can be projected onto each $\mathsf{S}^\times_j$ and will reach a goal and satisfy the projection of ψ.

4 LTL Decomposition

We now present a second admissible heuristic h^{dec}_ψ for a single PLTL constraint. This heuristic combines projection with a complementary relaxation of PLTL constraints, so it is assumed in this section that the underlying SSP and ψ have already been projected onto a subset of the SAS⁺ variables as in the previous section. We call this second relaxation *decomposition*, as it decomposes the progression formulae into sub-formulae. Decomposition over-estimates the probability of satisfying a formula in conjunctive normal form (CNF) by summing the probability of disjuncts and averaging the probability of conjuncts.

For the rest of this paper, we assume progression also converts formulae to CNF. Any LTL formula can be expanded to the form $\bigwedge_i (\alpha_i \vee \bigvee_j \phi_{ij})$ where each α_i is a finite disjunction of literals, and each ϕ_{ij} is an LTL formula in negation normal form prefixed by the next operator $\mathbf{X}$. During progression, each α_i can be evaluated against the current state interpretation, and we refer to the resulting form as CNF, to $\bigvee_j \phi_{ij}$ as a clause, and to ϕ_{ij} as an *X-literal*. While this CNF transformation results in an exponential blowup in formula size, it is only necessary in h^{dec}_ψ, and is used in conjunction with formula projection. In practice, projection simplifies the formula such that the exponential blowup doesn't significantly affect the heuristic's performance.

For convenience, we represent a formula in CNF as a set of sets; a set Ψ represents the formula $\bigwedge_{\Phi \in \Psi} \bigvee_{\phi \in \Phi} \phi$. The set of X-literals in a CNF formula Ψ is $\mathcal{D}(\Psi)$, called the decomposition of Ψ. It is convenient to define the set of X-literals which might arise from progression of a formula ψ. We denote this set $\Sigma(\psi)$, and it is the set of temporal subformulae of ψ prefixed by $\mathbf{X}$.

We denote the probability of satisfying a formula Ψ from a state s as $\mathsf{Pr}(\Psi|s)$. To estimate the probability $\mathsf{Pr}(\Psi|s)$ we observe the following inequalities:

$$\mathsf{Pr}(\Psi|s) \le \textstyle\sum_{\Phi \in \Psi} \mathsf{Pr}(\Phi|s)/|\Psi| \qquad (1)$$

$$\mathsf{Pr}(\Phi|s) \le \textstyle\sum_{\phi \in \Phi} \mathsf{Pr}(\phi|s) \qquad (2)$$

$$\mathsf{Pr}(\phi|s) = \max_{a \in \mathsf{A}(s)} \sum_{s' \in \mathsf{S}} P(s'|s,a) \mathsf{Pr}(\mathrm{prog}(s', \phi)|s') \qquad (3)$$

Applied recursively, these inequalities find an over-estimate for the probability while only optimising actions in states with X-literal modes in $\Sigma(\psi)$.

To encode this optimisation in our LP framework, we use a flow network which exhibits duplication of flow. The variables are occupation measures $x_{s,\phi,a}$ representing flow leaving the pair $\langle s, \phi \rangle$ via action a. Under SSP dynamics with progression, the flow along $x_{s,\phi,a}$ would be shared between the pairs $\langle s'$,prog$(s', \phi)\rangle$ such that s' is reachable via a, but

[5] It is possible that for a fringe state the projection of that state is not present in an LP, in which case that LP is extended to include this state and all projected states reachable from it.

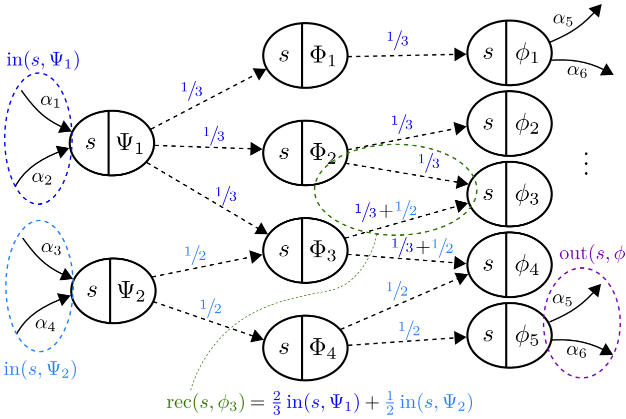

Figure 1: An example of flow from multiple states labelled with CNF formulae being redistributed to decomposed modes. Dotted lines denote the movement of flow by redistribution, and solid lines denote occupation measures.

instead is duplicated and redirected to pairs $\langle s', \phi' \rangle$ for $\phi' \in \mathcal{D}(\text{prog}(s', \phi))$. Let $\mathbf{I}_{\Psi,\phi} = |\{\Phi \in \Psi | \phi \in \Phi\}|$ be the number of occurrences of ϕ in Ψ. We define the functions:

$$\begin{aligned}
\text{in}(s, \Psi) &\equiv \sum_{\substack{s' \in \mathsf{S}, a \in \mathsf{A}(s') \\ \phi \in \Sigma(\psi) : \text{prog}(s,\phi) = \Psi}} P(s|s', a) x_{s',\phi,a} \\
\text{out}(s, \phi) &\equiv \textstyle\sum_{a \in \mathsf{A}(s)} x_{s,\phi,a} \\
\text{rec}(s, \phi) &\equiv \textstyle\sum_{\Psi : \phi \in \mathcal{D}(\Psi)} \frac{\mathbf{I}_{\Psi,\phi}}{|\Psi|} \text{ in}(s, \Psi)
\end{aligned}$$

The function in() represents the flow into states with CNF modes, rec() represents flow received from these to X-literal modes, and out() represents the flow leaving one of these state-X-literal pairs. Note that rec() represents both Eqs. (1) and (2) by splitting flow evenly between clauses and subsequently *duplicating* flow to each X-literal. See Fig. 1 for an example of this flow redistribution.

Flow in this network sinks at pairs $\langle s, \Psi \rangle$ not only where $s \in \mathsf{G}$ but also if $\Psi = \top$ or $\Psi = \bot$, as there is no decomposition for $\top$ and $\bot$. Let the set of these sinks be $\mathcal{F}$. We maximise flow into the accepting sinks, i.e., $\langle s, \Psi \rangle \in \mathsf{T}$ or $\Psi = \top$. Let the set of these pairs be $\hat{\mathsf{G}} \subset \mathcal{F}$. The amount of flow reaching the accepting sinks is represented by the variable $\textit{sink}_{\text{acc}}$.

The following linear constraints define the decomposition flow network, using only $O(|\mathsf{S} \times \Sigma(\psi) \times \mathsf{A}|)$ variables.

$$\begin{array}{lll}
\max & \textit{sink}_{\text{acc}} & \text{(LP2)} \\
\text{s.t.} & x_{s,\phi,a} \geq 0 \qquad \forall s \in \mathsf{S}, \phi \in \Sigma(\psi), a \in \mathsf{A}(s) & \text{(C8)} \\
& \textit{sink}_{\text{acc}} \leq 1 & \text{(C9)} \\
& \text{out}(s, \phi) - \text{rec}(s, \phi) \leq \displaystyle\sum_{\Psi : \langle s, \Psi \rangle \in \mathsf{F}} \mathbf{P}(s, \Psi) \times \frac{\mathbf{I}_{\Psi,\phi}}{|\Psi|} \quad \begin{array}{r} \forall s \in \mathsf{S} \setminus \mathsf{G}, \\ \phi \in \Sigma(\psi) \end{array} & \text{(C10)} \\
& \textit{sink}_{\text{acc}} - \displaystyle\sum_{\langle s, \Psi \rangle \in \mathsf{F} \cap \hat{\mathsf{G}}} \mathbf{P}(s, \Psi) = \displaystyle\sum_{\langle s, \Psi \rangle \in \hat{\mathsf{G}}} \text{in}(s, \Psi) & \text{(C11)}
\end{array}$$

Here C10 requires that flow leaving a state must enter it, and the right hand side allows for flow to be sourced from fringe states. Note that the right hand side will be 0 when there is no fringe state feeding into the decomposed state/X-literal pair, i.e., $\nexists \langle s, \Psi \rangle \in \mathsf{F}$ with $\phi \in \mathcal{D}(\Psi)$. This constraint is an inequality so that flow can be leaked from anywhere in the network, a technique previously used to solve SSPs with dead ends in (Trevizan, Teichteil-Königsbuch, and Thiébaux 2017). C11 defines $\textit{sink}_{\text{acc}}$, taking into account fringe states which project to the goal.

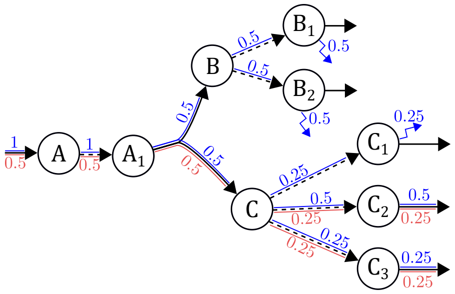

Figure 2: An example of flow synchronised between both networks. The blue/red labels above/below transitions show flow in the primary/secondary network.

5 Integration with PLTL-dual

The state-of-the-art heuristic search algorithm for solving MO-PLTL SSPs is PLTL-dual (Baumgartner, Thiébaux, and Trevizan 2018). We integrate both h^{pom}_{ψ} and h^{dec}_{ψ} with PLTL-dual to evaluate their performance. PLTL-dual interfaces with heuristics in a unique way, iteratively solving LPs representing progressively larger subsets of the state space, and including the heuristic computation in this same LP. In this way, the heuristic is computed for all fringe states at once, simultaneously with finding the shortest path.

The vast majority of the information extracted from heuristics in PLTL-dual is via tying constraints. For estimating the cost, PLTL-dual maintains an instance of h^{pom} (Trevizan, Thiébaux, and Haslum 2017), and this is tied to the heuristic for each constraint, so actions necessary to satisfy a constraint are used in the cost heuristic also.

While h^{pom}_{ψ} can be tied directly with PLTL-dual, tying h^{dec}_{ψ} first requires the introduction of the concept of flow retracing.

5.1 Flow Retracing

To over-estimate the probability, the network in the previous section (which we call the primary network) finds a probabilistic path through a relaxation of MO-PLTL SSP dynamics. Because of flow duplication, occupation measures in the primary network don't correspond to the original dynamics, compromising the use of tying constraints. To bridge this gap, we introduce a secondary network which retraces flow that reached accepting states without duplication. This can be thought of as finding a probability distribution on paths through the primary network to accepting states, as illustrated in Fig. 2. The secondary network is represented by further constraints which find this probability distribution concurrently with the optimisation of the primary network. It is this secondary network that we tie to h^{pom} in PLTL-dual.

Fig. 2 illustrates the secondary network tracing flow through the primary network. Since there is no accepting path from state B to the goal, primary flow into it is leaked after being redistributed. The probabilistic action out of A_1 lets only 0.5 units of flow into state C, all of which eventually reaches accepting goal states. The flow entering C_1 may be leaked even if there is a path to an accepting goal state, as the secondary network flow is bottlenecked by the 0.5 units of primary flow from A_1 to C. As such, the secondary network traces only 0.5 units of flow through the primary network.

The secondary network uses an all-outcomes determinisation (Yoon, Fern, and Givan 2007), allowing flow to "choose" which outcome of an action it follows, and which X-literal to be distributed to. The variables in the secondary network are occupation measures for both states with CNF modes and those with X-literals. These are denoted $y_{s,\Psi,\phi}$ and $y_{s,\phi,a,s'}$ respectively. The first represents flow being redistributed in the state s from the mode Ψ to ϕ, the second represents flow leaving s with ϕ via action a and reaching the state s'.

For convenience we define the following functions for the secondary network. These are self-explanatory, but have a notable distinction between X-literal modes and CNF modes, similarly to the primary network:

$$\mathrm{in}_2(s,\Psi) = \sum_{\substack{s'\in\mathsf{S},\phi\in\Sigma(\psi),a\in\mathsf{A}(s'):\\ \Psi=\mathrm{prog}(s,\phi)\wedge P(s|s',a)>0}} y_{s',\phi,a,s} \qquad \mathrm{out}_2(s,\phi) = \sum_{\substack{a\in\mathsf{A}(s),s'\in\mathsf{S}:\\ P(s|s',a)>0}} y_{s,\phi,a,s'}$$

$$\mathrm{in}_2(s,\phi) = \sum_{\Psi:\phi\in\mathcal{D}(\Psi)} y_{s,\Psi,\phi} \qquad \mathrm{out}_2(s,\Psi) = \sum_{\phi\in\mathcal{D}(\Psi)} y_{s,\Psi,\phi}$$

The secondary network retraces flow reaching accepting state pairs in $\hat{\mathsf{G}}$. As not all the flow reaching the fringe necessarily can leave the secondary network through these states, the flow entering it is similarly limited. We add the variables $i_{s,\Psi}$ for each fringe state to represent the amount of flow entering the secondary network from that state. The constraints for flow through the secondary network are:

$$y_{s,\phi,a,s'} \geq 0 \qquad \forall s,s'\in\mathsf{S},\phi\in\Sigma(\psi),a\in\mathsf{A}(s):P(s'|s,a)>0 \quad \text{(C12)}$$

$$y_{s,\Psi,\phi} \geq 0 \qquad \forall s\in\mathsf{S},\Psi\in 2^{2^{\Sigma(\psi)}},\phi\in\mathcal{D}(\Psi) \quad \text{(C13)}$$

$$0 \leq i_{s,\Psi} \leq 1 \qquad \forall\langle s,\Psi\rangle \in \mathrm{F}\setminus\mathcal{F} \quad \text{(C14)}$$

$$i_{s,\Psi} \leq \mathbf{P}(s,\Psi) \qquad \forall\langle s,\Psi\rangle \in \mathrm{F}\setminus\mathcal{F} \quad \text{(C15)}$$

$$\mathrm{out}_2(s,\Psi) - \mathrm{in}_2(s,\Psi) = i_{s,\Psi} \qquad \forall\langle s,\Psi\rangle \in \mathrm{F}\setminus\mathcal{F} \quad \text{(C16)}$$

$$\mathrm{out}_2(s,\Psi) - \mathrm{in}_2(s,\Psi) = 0 \qquad \forall\langle s,\Psi\rangle\in\mathsf{S}\times 2^{2^{\Sigma(\psi)}}\setminus(\mathrm{F}\cap\mathcal{F}) \quad \text{(C17)}$$

$$\mathrm{out}_2(s,\phi) - \mathrm{in}_2(s,\phi) = 0 \qquad \forall\langle s,\phi\rangle\in\mathsf{S}\times\Sigma(\psi) \quad \text{(C18)}$$

$$\textstyle\sum_{\langle s,\Psi\rangle\in\hat{\mathsf{G}}} \mathrm{in}_2(s,\Psi) = \sum_{\langle s,\Psi\rangle\in\mathrm{F}} i_{s,\Psi} \quad \text{(C19)}$$

The input variables $i_{s,\Psi}$ must not exceed the actual flow entering from the associated fringe state (C15), and the flow entering the network must be the same as the flow leaving it (C19). Similarly, the flow entering each state must equal the flow leaving it (C17, C18), allowing for the flow entering the network at fringe states (C16). Note that for C13 and C17, only the variables for the set of reachable CNF formulae need to be generated, rather than the full mode set $2^{2^{\Sigma(\psi)}}$.

As well as secondary network's constraints (C12-C19), we add constraints between the two networks, upper-bounding the secondary network relative to the primary network. Flow along actions is restricted by C21, and flow redistributed in decomposition is upper bounded by C22 and C23, where the right hand side of C22 takes into account flow entering the network from the fringe. C20 forces the flow leaving the secondary network to match the flow leaving the primary network. The combination of C11, C19 and C20 make the flow entering each network and leaving each network via pairs in $\hat{\mathsf{G}}$ identical.

$$\textstyle\sum_{\langle s,\Psi\rangle\in\hat{\mathsf{G}}} \mathrm{in}(s,\Psi) = \sum_{\langle s,\Psi\rangle\in\hat{\mathsf{G}}} \mathrm{in}_2(s,\Psi) \quad \text{(C20)}$$

$$y_{s,\phi,a,s'} - x_{s,\phi,a} \times P(s'|s,a) \leq 0 \qquad \forall y_{s,\phi,a,s'} \quad \text{(C21)}$$

$$y_{s,\Psi,\phi} - \mathrm{in}(s,\Psi) \times \frac{\mathbf{I}_{\Psi,\phi}}{|\Psi|} \leq \sum_{\Psi:\langle s,\Psi\rangle\in\mathrm{F}} \mathbf{P}(s,\Psi) \times \frac{\mathbf{I}_{\Psi,\phi}}{|\Psi|} \qquad \begin{array}{r}\forall\langle s,\Psi\rangle\in\mathrm{F},\\ \phi\in\mathcal{D}(\Psi)\end{array} \quad \text{(C22)}$$

$$y_{s,\Psi,\phi} - \mathrm{in}(s,\Psi) \times \tfrac{\mathbf{I}_{\Psi,\phi}}{|\Psi|} \leq 0 \qquad \forall\langle s,\Psi\rangle\in\mathsf{S}\times 2^{2^{\Sigma(\psi)}}\setminus\mathrm{F},\phi\in\mathcal{D}(\Psi) \quad \text{(C23)}$$

5.2 Tying Constraints

As mentioned above, PLTL heuristics in PLTL-dual are tied to the cost heuristic which consists of a set of projections, one of each variable v, and ignores the PLTL constraints.

The cost heuristic can be tied to h_ψ^{pom} using constraints similar to C7. The distribution $\mathbf{P}$ is then defined by the flow into the fringe states in PLTL-dual and the cost and PLTL heuristics are computed concurrently with the path optimisation.

Tying h_ψ^{dec} to the cost heuristic is also relatively straightforward. Let $x_{d,a}$ be the occupation measure for the action a and the value $d\in\mathsf{D}_v$ of the projection onto some variable v used by the cost heuristic. Each projection in h_ψ^{dec} is then tied this the projection onto v using the tying constraints below:

$$\sum_{\substack{s,s'\in\mathsf{S},\phi\in\Sigma(\psi):\\ P(s|s',a)>0}} y_{s,\phi,a,s'} \leq \sum_{d\in\mathsf{D}_v} x_{d,a} \qquad \forall a\in\mathsf{A} \quad \text{(C24)}$$

The choice of v in C24 doesn't matter, as all projections used by the cost heuristic are all tied together through equality constraints and will yield the same result. Also, note that the inequality in C24 forces actions necessary to satisfy the constraint to be taken in each of the variable projections of the cost heuristic, but not vice versa. This is because, the network for each given PLTL constraint reaches its sink as soon as an accepting state is reached for that constraint, rather than continuing executing actions to reach the goal.

5.3 Admissibility of h_ψ^{dec} in PLTL-dual

We provide a sketch of the proof of the admissibility of h_ψ^{dec} when integrated with PLTL-dual. Let $\hat{x}$ be a solution to LP1, inducing a valid policy π and let $z = \mathsf{Pr}(\psi \mid s_0, \pi)$. W.l.o.g., we show that $h_\psi^{\mathrm{dec}}(\langle s_0, \Psi_0\rangle) \geq z$ (the case of other extended states $\langle s, \Psi\rangle$ can be handled by using them as initial state and adjusting z appropriately). Also, let $out(s, \Psi \mid \pi)$ be the flow out of $\langle s, \Psi\rangle$ in our solution $\hat{x}$.

Suppose for the sake of contradiction that $h_\psi^{\mathrm{dec}}(\langle s_0, \Psi_0\rangle) < z$. The proof sketch is as follows: We first construct a solution to the secondary network from $\hat{x}$ satisfying C12-C19, then a solution for the primary network satisfying C8-C11 and show that both networks satisfy constraints C20-C23.

To construct the solution to the secondary network, let $T_{s_0,\Psi_0,\pi}$ be the set of finite trajectories $t =$

$\langle s_0, \Psi_0\rangle, ..., \langle s_n, \Psi_n\rangle$ from $\langle s_0, \Psi_0\rangle$ reachable under π, such that $t \models \Psi_0$. By the semantics of progression, for each trajectory t there exists at least one sequence of sets of LTL formulae $S_0^t, ..., S_n^t$ (called satisfying assignments) such that

$$
\begin{aligned}
&S_i^t \subseteq \mathcal{D}(\Psi_i),\\
&\forall \Phi \in \Psi_i, S_i^t \cap \Phi \neq \emptyset,\\
&\forall \phi_{ij}^t \in S_i^t, \forall \Phi_k \in \text{prog}(\phi_{ij}^t, s_{i+1}), |S_{i+1}^t \cap \Phi_k| = 1 \text{ and}\\
&\forall \phi_{ij}^t \in S_i^t, t[i...n] \models \phi_{ij}^t
\end{aligned}
$$

where $t[i...n]$ is the subsequence of t starting at index i. These sets represent, at each step, the literals in the formula which will be satisfied by the remainder of the trajectory. We can iteratively assign each ϕ_{ij}^t several weights w_{ijk}^t, where $w_{0j0}^t = \frac{\mathbf{I}_{\Psi_0, \phi_{0j}^t}}{|\Psi_0|}$, and $w_{ijk}^t = \frac{\mathbf{I}_{\text{prog}(\phi_{(i-1)k}^t, s_i^t), \phi_{ij}^t}}{|\text{prog}(\phi_{(i-1)k}^t, s_i^t)|} \cdot \sum_l w_{(i-1)kl}^t$. These weights have the property that $\forall i \ \sum_{j,k} w_{ijk}^t = 1$.

We define all $y_{s,\phi,a,s'}$ and $y_{s_0,\Psi_0,\phi}$ as follows:

$$y_{s,\phi,a,s'} := \sum_{t \in T_{s_0,\Psi_0,\pi}} \mathsf{Pr}(t) \cdot \Big(\sum_{\substack{i,j,k \text{ s.t. } s_i^t = s,\\ \phi_{ij}^t = \phi, a_i^t = a, s_{i+1}^t = s'}} w_{ijk}^t \Big)$$

$$y_{s_0,\Psi_0,\phi} := \sum_{t \in T_{s_0,\Psi_0,\pi}} \mathsf{Pr}(t) \cdot \Big(\sum_{j \text{ s.t. } \phi = \phi_{0j}^t} w_{0j0}^t \Big)$$

and all other $y_{s,\Psi,\phi}$ as:

$$y_{s,\Psi,\phi} := \sum_{t \in T_{s_0,\Psi_0,\pi}} \mathsf{Pr}(t) \cdot \Big(\sum_{\substack{i,j,k \text{ s.t. } s_i^t = s,\\ \text{prog}(\phi_{(i-1)k}^t, s_i^t) = \Psi, \phi_{ij}^t = \phi}} w_{ijk}^t \Big)$$

Under this construction, as the weights at each step sum to 1, then the flow entering accepting goal states in the secondary network is $\sum_{t \in T_{s_0,\Psi_0,\pi}} \mathsf{Pr}(t) = z$.

For the primary network, we construct a solution from the trajectories $T'_{s_0,\Psi_0,\pi}$, which is the set of trajectories induced by π, *not* conditioned on satisfying Ψ_0. For a trajectory t, for each $\phi_{ij}^t \in \mathcal{D}(\Psi_i^t)$ we assign a weight v_{ij}^t. Note that unlike weights w_{ijk}^t, these are not limited to a set of satisfying assignments. We assign $v_{0j}^t = \frac{\mathbf{I}_{\Psi_0, \phi_{0j}^t}}{|\Psi_0|}$, and $v_{ij}^t = \sum_k \frac{\mathbf{I}_{\text{prog}(\phi_{(i-1)k}^t, s_i^t), \phi_{ij}^t}}{|\text{prog}(\phi_{(i-1)k}^t, s_i^t)|} \cdot v_{(i-1)k}^t$. These weights have the property that, for $\phi_{ij} \in S_i$, $v_{ij}^t \geq \sum_k w_{ijk}^t$. We then assign

$$x_{s,\phi,a} := \sum_{t \in T'_{s_0,\Psi_0,\pi}} \mathsf{Pr}(t) \cdot \Big(\sum_{\substack{i,j \text{ s.t. } s_i^t = s,\\ \phi_{ij}^t = \phi, a_i^t = a}} v_{ij}^t \Big)$$

with an exception for x_{s,ϕ,a_g} which route flow into accepting states. These variables are set to

$$x_{s,\phi,a_g} := \frac{y_{s,\phi,a_g,g}}{\mathsf{Pr}(g \mid s, a_g)}$$

where g is a goal state. Due to this, the primary network has exactly z units of flow entering accepting states.

Note that tying constraints are consistent with this construction when π is projected onto other cost constraints. Now that we have a valid flow for both primary and secondary networks, we need to show that they satisfy constraints C20–C23, i.e., the primary network upper-bounds the secondary network. Consider an action a from a state $\langle s, \phi\rangle$. By construction, outcomes of a must be taken proportionately to $P(s' \mid s, a)$ by the secondary network except when the associated trajectories are not in $T_{s_0,\Psi_0,\pi}$. All trajectories in $T'_{s_0,\Psi_0,\pi}$ are included in the primary network, hence satisfying the upper bound in C21. From this we have $\text{in}(s, \Psi) \geq \text{in}_2(s, \Psi)$, and as weights are distributed equivalently in both the primary and secondary network, constraints C22 and C23 are also satisfied. Lastly, C20 is true by construction, as both networks admit exactly z flow into accepting states which contradicts our assumption that $h_\psi^{\text{dec}}(\langle s_0, \Psi_0\rangle) < z$; therefore h_ψ^{dec} is admissible.

6 Experimental Results

The heuristics were evaluated in comparison with the NBA heuristic h^{BA} (Baumgartner, Thiébaux, and Trevizan 2018) and the trivial heuristic ($h(\mathbf{P}) = 1$), as well as PRISM. As the results for the 2019 Comparison of Tools for the Analysis of Quantitative Formal Models (Hahn et al. 2019) shows, PRISM is still one of the fastest model checkers available. We used the default options of PRISM and the "-lp" flag to use their LP approach to MO-PLTL SSPs because, without this flag, PRISM was unable to solve any of our benchmarks. The experiments were ran on an Intel i7-7700@3.6GHz using Gurobi 8.1.1 on a single thread and a 20mins and 4Gb cutoff.

For evaluation, we use the Factory and Wall-e domains from (Baumgartner, Thiébaux, and Trevizan 2018), and a new domain called Priority Search. All domains include at least one non co-safe and one non-safe formula. The problems in our experiments are represented in PPDDL (Younes et al. 2005) and translated to probabilistic SAS$^+$ using Fast Downward (Helmert 2006). For PRISM, the probabilistic SAS$^+$ problems are translated to PRISM's multi-valued language. Both PPDDL and PRISM versions of the problems are available at https://gitlab.com/fwt/mo-pltl-ssps-benchmarks.

The Priority Search domain is based on the Search and Rescue (SAR) domain (Trevizan, Thiébaux, and Haslum 2017) and, in both domains, the agent controls a robot locating missing victims of a disaster in an $n \times n$ grid. Each position is randomly initialised as possibly containing a victim with probability p. As in the SAR domain, the actions represent the agent moving in the environment and exploring the current position; however, the fuel resource from SAR is omitted to avoid confounding effects in the experiment since h^{pom} can handle them with ease which could be seen as an unfair advantage against the baselines. The constraints in our domain are also different from SAR which does not have any PLTL constraints. The first constraint requires that eventually all unknown locations must be searched. A contiguous line of $n - 1$ locations is randomly initialised as the "danger zone". The second and third constraints are that locations in the danger zone must be searched first, and with probability at least 0.8, the robot must not stay inside the danger zone for more than 2 steps.

The results, as the aggregate of multiple runs for each parameterisation of each domain, are presented in Fig. 3 for

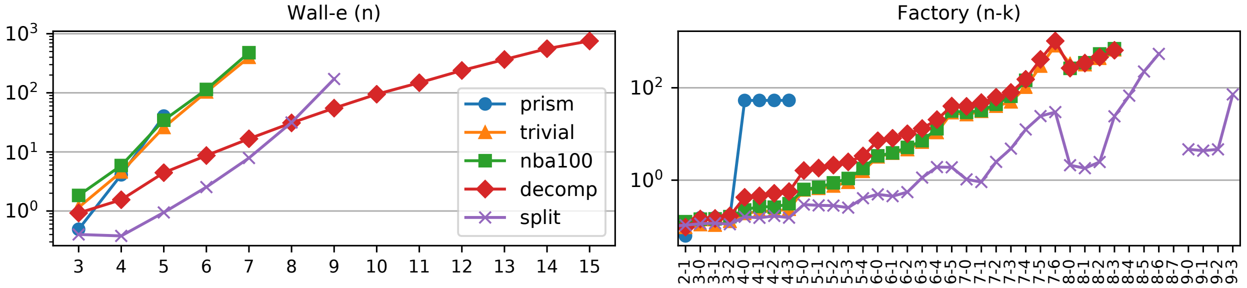

Figure 3: The solution time in seconds averaged over 10 runs for each problem.

n	p	PRISM	Trivial Heuristic	h^{BA}	h^{dec}_{ψ}	h^{dec}_{ψ}-*pos*	h^{pom}_{ψ}	h^{pom}_{ψ}-*pos*
4	0.25	**100%** **1.72±0.03**	**100%** 1.82±0.17	75% 652±158	**100%** 5.49±0.37	**100%** 6.69±1.35	**100%** 4.25±0.30	**100%** 4.15±0.82
	0.50	**100%** 57.6±1.94	**100%** 74.1±24.9	**100%** 272±88.8	**100%** 167±53.8	**100%** 60.4±23.2	**100%** 141±50.6	**100%** **32.3±12.6**
	0.75	0% n.a.	0% n.a.	0% n.a.	0% n.a.	55% 624±222	0% n.a.	**75%** **265±143**
5	0.25	**100%** **34.7±1.68**	**100%** 46.0±130	**100%** 127±36.2	**100%** 128±30.0	**100%** 75.2±15.6	**100%** 91.6±23.3	**100%** 40.4±11.0
	0.50	0% n.a.	0% n.a.	0% n.a.	0% n.a.	30% 600±457	0% n.a.	**70%** **330±147**
	0.75	0% n.a.	0% n.a.	0% n.a.	0% n.a.	0% n.a.	0% n.a.	**15%** **503±1124**

Table 1: Results for Priority Search(n,p) problems averaged over 20 runs for each problem. The results are reported as "X Y±Z" where X is the percentage of problems successfully solved and Y and Z are the average and its 95% conf. interval for the cpu-time over solved problems. Best values for each problem is highlighted.

Wall-e and Factory, and in Table 1 for Priority Search. The plots show the average time over 10 runs to solve problems (with its 95% confidence interval) in the Factory and Wall-e domains. The graphs use a log scale, and we omit points which had less than 100% coverage. Table 1 shows the results for 6 parameterisations of the Priority Search domain with 20 runs per entry. Since the algorithms considered compute the optimal solution and do not rely on sampling, 10 runs is enough to account for minor sources of randomisation, e.g., tie breaking. The only exception is the random choice of combinations used by h^{pom}_{ψ} (Section 3.2); however, for both wall-e and factory domains, there is only one possible combination and, for priority search domain, the 20 runs account for the multiple possible combinations. Table 2 presents relevant statistics of selected problems for each approach, e.g., size of the SSP, LPs, and automata, namely Deterministic Rabin Automata (DRA) for PRISM and NBA for PLTL-dual with the h^{BA} heuristic.

In the Wall-e domain, either h^{pom}_{ψ} or h^{dec}_{ψ} dominates all the other planners for $n > 3$ and they are at least one order of magnitude faster than the others planners for the largest problems solved by them ($n \in \{6, 7\}$). h^{pom}_{ψ} has the best performance up to $n = 8$ because it provides better guidance than the other heuristics: h^{pom}_{ψ} explored on average 31.3% (95%ci: ±0.02) and 8.2% (95%ci: ±0.03) of the states explored by h^{dec}_{ψ} and h^{BA} respectively for $n \geq 4$. However, h^{pom}_{ψ} is unable to scale up as well as h^{dec}_{ψ}. For instance, for $n = 7$, h^{pom}_{ψ} uses almost the double of LP variables as h^{dec}_{ψ} for heuristic encoding (see Table 2). This is because the Wall-e domain has at most 2 state variables in each constraint and, more importantly, only a single state variable in its largest constraint, making the h^{pom}_{ψ} constraint relaxation ineffective. For $n = 16$, h^{dec}_{ψ} solved 9 out of 10 runs in 18m40s and exceeded the 20mins cutoff in a single run while all runs exceeded the cutoff for $n = 17$.

For Factory, h^{pom}_{ψ} dominates all the other algorithms for anything but small problems and is 1 to 2 orders of magnitude faster than the other planners for $n \in \{7{,}8\}$ and all k. Moreover, h^{pom}_{ψ} is the only planner able to scale up to $n = 8, k > 4$ and $n = 9$. As in Wall-e, the dominance of h^{pom}_{ψ} is because it is more informative than the other heuristics: it explored on average 44.5% (95%ci: ±0.09) of the states explored by the second best planner for $n > 4$ and all k. This can also be observed in the size of the LP solved using h^{pom}_{ψ}, for instance, in problem 8-3, h^{pom}_{ψ} uses 50% less LP variable to encode its heuristic than h^{dec}_{ψ} and its last LP solved is 20% of the size of h^{dec}_{ψ}'s last LP. Notably, it was slower to apply h^{dec}_{ψ} than the trivial heuristic in factory, which can be attributed to decomposition not being informative enough to make up for

		Wall-e (n)			**Factory** (n-k)			**Priority Search** (n, p)		
	problem	5	7	15	4-3	8-3	9-3	4, 0.5	5, 0.25	5, 0.50
SSP	Size of S	90	182	870	256	65K	262K	4K	3K	212K
	Size of A	315	671	3K	1K	736K	3M	38K	31K	2M
PRISM	Size of largest DRA	2,857	–	–	29,979	–	–	256	128	–
	# LP vars. used	80,920	–	–	3,572	–	–	112,863	92,467	–
h^{BA}	largest NBA size	48	256	–	13	137	266	512	256	–
	# LP vars. for heur.	8,913	3,498	–	418	224	248	2,265	3,921	–
	# vars. used in last LP	18,474	33,337	–	1,552	42,513	–	47,878	43,885	–
	% time solving LPs	90.1%	90.1%	–	41.9%	96.5%	–	59.9%	72.9%	–
trivial h.	# vars. used in last LP	10,681	33,492	–	1,070	40,836	–	33,613	31,125	
	% time solving LPs	82.3%	90.4%	–	44.3%	95.4%	–	83.1%	77.4%	–
h_ψ^{dec}	# LP vars. for heur.	5,454	13,412	116,444	785	4,683	6,110	9,264	13,943	23,822
	# vars. used in last LP	9,533	20,134	145,071	1,965	36,891	–	27,457	32,188	68,421
	% time solving LPs	70.9%	74.6%	81.4%	58.5%	96.7%	–	85.6%	88.4%	89.6%
h_ψ^{pom}	# LP vars. for heur.	5,230	26,334	–	446	2,022	2,596	15,447	21,719	40,834
	# vars. used in last LP	7,376	29,427	–	972	7,383	28,435	28,879	35,515	78,110
	% time solving LPs	19.3%	30.7%	–	16.1%	81.6%	84.3%	74.2%	74.1%	85.5%

Table 2: Statistics for each approach in selected problems. For priority search, the statistic are for h_ψ^{dec}-*pos* and h_ψ^{pom}-*pos*.

its computation time. For instance, h_ψ^{dec} expanded only 8% fewer states than with the trivial heuristic on average, as the single directional tying constraints (C24) don't capture the bidirectional relationship between machines and production.

Lastly, for the Priority Search domain, the difference between h_ψ^{dec} and h_ψ^{pom} is not statistically significant and they were dominated by PRISM for $n = 4, p \in \{0.25, 0.5\}$ and $n = 5, p = 0.25$. h_ψ^{dec} and h_ψ^{pom} underperform in this domain because the choice of variables made by the process in Section 3 is sub-optimal. This can be verified by manually including the robot's position in the projections which we refer to as h_ψ^{pom}-*pos* and h_ψ^{dec}-*pos*. Considering h_ψ^{pom}-*pos* and h_ψ^{dec}-*pos*, we have that h_ψ^{pom}-*pos* is statistically tied with PRISM in the small problems and dominates all other planners for large instances ($n = 4, p = 0.75$ and $n = 5, p \geq 0.5$). While statistically tied on time for small problem, h_ψ^{pom}-*pos* expanded 15.1% (95%ci: ±0.06) of the states visited by PRISM on average, thus solving much smaller LPs, e.g., for $n = 5, p = 0.25$, its final LP's size is on average 39% of that of PRISM's single LP. This advantage allows h_ψ^{pom}-*pos* to scale up to larger problems than PRISM. Moreover, h_ψ^{dec}-*pos* is slower than PRISM for small problem but it is still capable to scale up to larger problems than PRISM. We can also see that h_ψ^{pom}-*pos* provides better guidance than h_ψ^{dec}-*pos*, for instance h_ψ^{pom}-*pos* expanded 77.6% (95%ci: ±5.65) of the states expanded by h_ψ^{dec}-*pos* on problem $n = 5, p = 0.25$. Note however that PRISM is a more general tool that can solve a larger class of problems than MO-PLTL SSPs.

7 Conclusion, Related and Future Work

We presented new admissible heuristics for probabilistic planning with MO-PLTL constraints, and showed they compared favourably to the only other heuristic available for these problems (Baumgartner, Thiébaux, and Trevizan 2018). The strength and novelty of our heuristics lie in principled ways of choosing sets of variables on which to project LTL formulae and relaxing the computation of their probabilities. These contributions are enabled by progression, showing promise for progression-based approaches to PLTL heuristics.

In related work, Lacerda et al. (2015) define a "task progression" metric that estimates the number of transitions required to reach an accepting state from a given state of a finite automaton for a co-safe LTL formula. This is then used in a multi-objective MDP, to reward the extent to which an LTL formula that cannot be satisfied by any policy has progressed towards an accepting state. The metric could be seen as an admissible heuristic for co-safe LTL, which is however not informed by the possible transitions of the environment.

Outside of probabilistic planning, there is a body of work on heuristics for deterministic planning with temporally extended goals and preferences expressed using LTL variants (Baier, Bacchus, and McIlraith 2009; Bienvenu, Fritz, and McIlraith 2011). Among those, (Bienvenu, Fritz, and McIlraith 2011) uses progression to optimistically evaluate formulae at fringe states, assuming the part of the formula that cannot be idled to false in the current state is true. In addition to being designed for a different problem where satisfaction probability needs to be evaluated, our projection heuristic can be more informative, as it is only optimistic over a subset of the formula's variables. The remainder of heuristic search approaches to planning with LTL constraints compiles LTL and finite LTL variants into various types of automata whose description can directly be incorporated in the factored planning problem descriptions. They are then handled using standard heuristics that are not LTL-aware (Rintanen 2000; Baier and McIlraith 2006; Edelkamp 2006; Torres and Baier 2015; Camacho et al. 2017).

Our future work includes decreasing the large number of variables introduced to retrace accepting flow, since they are greatly responsible for the overhead in h_ψ^{dec}. We also plan to experiment with techniques to split automata (Camacho et al. 2018) to improve PLTL-dual and the heuristics. Finally, we would like to extend the heuristic search approach to deal with more expressive logics (Baumgartner, Thiébaux, and Trevizan 2017) and partially observable domains (Santana, Thiébaux, and Williams 2016; Walraven and Spaan 2018).

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was funded by the Australian Research Council Discovery Project grant DP180103446 “On-line Planning for Constrained Autonomous Agents in an Uncertain World”.

References

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

Babiak, T.; Kretínský, M.; Rehák, V.; and Strejcek, J. 2012. LTL to Büchi Automata Translation: Fast and More Deterministic. In *Proc. Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, 95–109.

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

Bäckström, C. 1992. Equivalence and Tractability Results for SAS+ Planning. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, 126–137.

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

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A heuristic search approach to planning with temporally extended preferences. *Artif. Intell.* 173(5-6): 593–618.

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 (AAAI)*, 788–795.

Bauer, A.; and Haslum, P. 2010. LTL Goal Specifications Revisited. In *Proc. European Conf. on Artificial Intelligence (ECAI)*, 881–886.

Baumgartner, P.; Thiébaux, S.; and Trevizan, F. W. 2017. Tableaux for Policy Synthesis for MDPs with PCTL* Constraints. In *Proc. Int. Conf. on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX)*, 175–192.

Baumgartner, P.; Thiébaux, S.; and Trevizan, F. W. 2018. Heuristic Search Planning With Multi-Objective Probabilistic LTL Constraints. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, 415–424.

Bienvenu, M.; Fritz, C.; and McIlraith, S. A. 2011. Specifying and computing preferred plans. *Artif. Intell.* 175(7-8): 1308–1345.

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

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018. Finite LTL Synthesis as Planning. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 29–38.

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 (AAAI)*, 3716–3724.

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 (IJCAI)*, 854–860.

Edelkamp, S. 2006. On the Compilation of Plan Constraints and Preferences. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 374–377.

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, Communication and Software Systems (SFM)*, 53–113.

Hahn, E. M.; Hartmanns, A.; Hensel, C.; Klauck, M.; Klein, J.; Křetínský, J.; Parker, D.; Quatmann, T.; Ruijters, E.; and Steinmetz, M. 2019. The 2019 Comparison of Tools for the Analysis of Quantitative Formal Models. In *Proc. Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, 69–92.

Helmert, M. 2006. The Fast Downward planning system. *J. Artif. Intell. Res.* 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 161–169.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible Abstraction Heuristics for Optimal Sequential Planning. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 176–183.

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 (CAV)*, 585–591.

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 (ATVA)*, 5–22.

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 (IJCAI)*, 1587–1593.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-Based Heuristics for Cost-Optimal Planning. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 226–234.

Rintanen, J. 2000. Incorporation of Temporal Logic Control into Plan Operators. In *Proc. European Conf. on Artificial Intelligence (ECAI)*, 526–530.

Santana, P.; Thiébaux, S.; and Williams, B. C. 2016. RAO*: An Algorithm for Chance-Constrained POMDP's. In *Proc. AAAI Conf. on Artificial Intelligence (AAAI)*, 3308–3314.

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 (IJCAI)*, 1696–1703.

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 (UAI)*.

Trevizan, F. W.; Thiébaux, S.; and Haslum, P. 2017. Occupation measure heuristics for probabilistic planning. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 306–315.

Vardi, M. Y.; and Wolper, P. 1994. Reasoning About Infinite Computations. *Information and Computation* 115(1): 1–37.

Walraven, E.; and Spaan, M. T. J. 2018. Column Generation Algorithms for Constrained POMDPs. *J. Artif. Intell. Res.* 62: 489–533.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 352–359.

Younes, H. L.; Littman, M. L.; Weissman, D.; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *J. Artif. Intell. Res.* 24: 851–887.

Zilberstein, S.; Washington, R.; Bernstein, D. S.; and Mouaddib, A. 2001. Decision-Theoretic Control of Planetary Rovers. In *Proc. Int. Sem. on Advances in Plan-Based Control of Robotic Agents*, volume 2466 of *LNCS*, 270–289.