Multi-Objective Optimal Control with Safety as a Priority

Kendra Lesser, *Member, IEEE*, and Alessandro Abate, *Member, IEEE*

***Abstract*—This work develops a lexicographic approach to multi-objective optimal control on models for cyber-physical systems, encompassing in particular stochasticity, limited access to model variables (partial observations), and possibly hybrid (continuous and discrete) dynamics (with the finite state POMDP framework as a known special instance). The technique is showcased in two new case studies in the area of smart buildings. Technically, the main achievements of this work are: The application of the lexicographic framework to multi-objective optimization including quantitative probabilistic safety requirements, thus leading to a principled and scalable integration of correct-by-design synthesis for safety and optimal synthesis for performance; the novel extension of the lexicographic framework to partially-observed stochastic models with continuous (possibly hybrid) dynamics; the emphasis on computational aspects, including the use of compact and approximate representations of value functions combined with the quantification of error bounds on model abstractions.**

I. Introduction

Safety-critical systems require stringent guarantees that the control processes governing their operation do not lead to unsafe configurations. Smart-energy buildings, infrastructure networks, and electricity grids are all examples of systems that integrate control signals with underlying continuous physical processes, and for which failure to operate safely would be prohibitively costly. Rather than checking if safety specifications are met *after* a controller is designed (known as validation), one way to achieve safety guarantees is to enforce safety requirements during the control design process (called "correct-by-design" control), either by treating safety as a constraint that must be satisfied or by treating it as the objective of an optimal control problem.

Often such complex models of physical processes are subject to uncertainties, either due to inaccuracies in the model or to environmental influences (weather-related, delays, or presence of humans in the loop) that cannot be predicted exactly. Controllers further require information about the system, often exchanged through sensors that a) may not be ubiquitous, and cannot measure all necessary information, and b) may introduce noise, and hence inaccuracies, into their measurements. *Partially observable* stochastic models, such as finite state partially observable Markov decision processes (POMDPs) or partially observable stochastic hybrid systems (POSHS), with both discretely and continuously evolving states, provide a fitting and realistic modelling framework.

Kendra Lesser is with Verus Research, 6100 Uptown Blvd NE, Suite 260, Albuquerque, NM 87110 (e-mail kendra.lesser@verusresearch.net).

Alessandro Abate is with the Department of Computer Science, University of Oxford, Parks Road, OX1 3QD, Oxford, UK (e-mail Alessandro.Abate@cs.ox.ac.uk).

The inclusion of stochasticity leads to regarding safety probabilistically - it is difficult, when not overtly conservative, to guarantee it with certainty, that is with probability equal to one. If we want to *design* a safe controller, we could synthesise a controller that maximises the probability of safety, as done in [1] for fully observable stochastic hybrid systems, and in [2]–[4] for POSHS. The major drawback to this approach is that safety is rarely the *only* concern when designing controllers: stabilisation, tracking, or cost minimisation are just some other possible objectives of interest. We are thus led to a heterogeneous *multi-objective* optimisation setup, where we not only want to design a controller that maximises the safety of the system, but also takes into account these additional performance-related objectives. In order to tackle this synthesis goal, one option is to synthesise a permissive controller, which provides a set of control inputs that satisfy a given requirement (i.e. safety), so that some other objective can be optimised over the set of available inputs. This is done for finite state Markov decision processes (MDPs) in [5], and for partially observable (but otherwise deterministic) discrete event systems, in [6], [7], although none actually consider a second control objective. Another possibility is to treat safety as a constraint to be enforced in the optimisation of a different objective, as in [8], which specifically considers enforcement with probability one, and only for finite state MDPs. Multi-objective optimisation of stochastic games is considered in [9] for complex objectives, such as safety, although partial observability is not considered.

None of the above multi-objective approaches are easily extended to general partially-observable stochastic systems, which are more generally rarely considered in the context of verification and formal synthesis (notable exceptions are: [10] introducing verification of hidden Markov models, [11] considering synthesis over POMDPs for qualitative (yes/no) specifications, and [12] discussing decidability results for POMDPs).

Over partially observable models, we would further like the possibility to explore the trade-off between safety and other objectives, which is not possible when we require safety to be enforced with probability one. We therefore take a lexicographic optimisation approach, and apply it to the formulation of safety as a unique specification presented for POSHS in [2], [3]. Lexicographic optimisation utilises multiple objective functions and assigns a preference (or priority) to each. A set of control inputs is generated that produces outcomes within a tolerance from the optimal solution, and that subset is then used as the set of possible inputs over which the next objective is optimised, thus taking a hierarchical approach to controller synthesis. This technique was first introduced in [13], and more recently elaborated upon for MDPs in [14] and for POMDPs in [15]. In [15], however, results are only presented for finite state models, and some details are overlooked, so that the results are not applicable to complex objectives such as safety.

Our contributions are several. First, we extend the lexicographic optimisation framework to multi-objective optimisation with safety requirements (i.e. with dissimilar cost objectives), thus integrating correct-by-design synthesis and synthesis for optimal performance. Second, we apply the lexicographic framework to partially observable stochastic systems with an extension to models with continuous or hybrid dynamics. Third, we provide an approximate computational technique to solve the lexicographic optimisation problem based on point-based value iteration (PBVI) [16], and quantify error bounds for this approximation, as well as error bounds on the abstraction necessary to reduce stochastic continuous or hybrid systems to finite state POMDPs. Finally, throughout the article we apply our theoretical work on two case studies in the area of building automation systems.

II. Background

A. Notation

First, we provide a brief overview of the notation we shall use. We denote expected value by $\mathbb{E}$, and a generic probability measure by $\mathbb{P}$. A probability measure or expected value induced by a control policy π (to be defined later), is $\mathbb{P}^\pi$ or $\mathbb{E}^\pi$, respectively. For a space $\mathcal{S}$, s_n is an element of $\mathcal{S}$ and typically represents the state of a system at time n. We use $|\cdot|$ to denote either absolute value for $s \in \mathbb{R}$, or cardinality for a finite set Ω. The vector 1-norm is denoted $\|\cdot\|_1$. The complement of a set K is given by K^c, and the indicator function over K, $\mathbf{1}_K(s)$, evaluates to 1 if $s \in K$, and 0 otherwise.

B. POMDP Model

We consider systems that can be modelled as partially observable Markov decision processes (POMDPs). A standard POMDP has discrete states, actions, and observations, and is studied over an additive cost function.

Definition 1: A POMDP is a tuple $\mathcal{J} = (\mathcal{S}, \mathcal{U}, \mathcal{Y}, T, Y, R)$, where

1) $\mathcal{S}$ is a finite set of states
2) $\mathcal{U}$ is a finite set of possible control inputs
3) $\mathcal{Y}$ is a finite set of observations
4) $T : \mathcal{S} \times \mathcal{S} \times \mathcal{U} \to [0,1]$ is a state transition function that assigns a probability measure to state s_{n+1} given state s_n and control u_n for all n: $T(s_{n+1}|s_n, u_n)$
5) $Y : \mathcal{Y} \times \mathcal{S} \times \mathcal{U} \to [0,1]$ is an observation function that assigns a probability measure to observation y_n given state s_n and control u_{n-1} for all n: $Y(y_n|s_n, u_{n-1})$
6) $R : \mathcal{S} \to [0,1]$ is an initial probability measure over the state space $\mathcal{S}$: $R(s)$

In this model the state is in general not accessed, and is therefore not available for control. Instead, a controller has access to all past observations and control inputs, which are stored in an information vector $i_n = (u_0, \ldots, u_{n-1}, y_1, \ldots, y_n) \in \mathcal{I}_n = \mathcal{U}^n \times \mathcal{Y}^n$. A control input is thus selected via a policy π, which maps the available information to the set of possible controls. More precisely:

Definition 2: A policy π for a POMDP $\mathcal{J}$ over a time horizon N is a sequence of functions, $\pi = (\pi_0, \ldots, \pi_{N-1})$, such that $\pi_n : \mathcal{I}_n \to \mathcal{U}$.

We consider only non-randomised policies (each input i_n produces a single output u_n) and the set of all possible non-randomised policies is denoted by Π. Typically, a control policy is obtained as the argument that minimises an expected sum of costs associated with the state and control input:

$$\pi^* = \arg\min_{\pi \in \Pi} \mathbb{E}^\pi \left[\sum_{n=0}^{N} C(s_n, u_n) \middle| R \right]. \quad (1)$$

The execution of a POMDP proceeds as follows. At time $n = 0$, state s_0 is produced from the initial distribution $R : s_0 \sim R(\cdot)$. At each subsequent time step $n > 0$, the state is s_n. An observation y_n is produced according to $y_n \sim Y(\cdot|s_n, u_{n-1})$. This observation is added to the list of past observations and control inputs to produce $i_n = (u_0, \ldots, u_{n-1}, y_1, \ldots, y_n)$. A new control input is chosen according to $u_n = \pi_n(i_n)$ (obtained solving (1)), and cost $C(s_n, u_n)$ is accrued. Then the state evolves according to $s_{n+1} \sim T(\cdot|s_n, u_n)$. At the final time step, when no control input is needed, the cost function $C(s_N, u_N)$ is modified to $C(s_N, 0)$.

1) Case Study: Optimised Boiler Maintenance: To illustrate how a POMDP can model a physical system, consider a simplified maintenance problem for a single boiler that heats a building[1]. The boiler operates with a certain level of efficiency that is captured through a degradation model. The degradation level slowly increases in time as a result of normal use, which we model using a stochastic difference equation as

$$s_{n+1} = s_n + v_n, \quad (2)$$

with s_n denoting the degradation level at discrete time step n, and v_n a random variable taking only non-negative values from the set $\mathcal{V}(s_n)$, i.e. $v_n \in \mathcal{V}(s_n)$ for all n, where the set is a function of the current state s_n. The probability that $v_n = \bar{v} \in \mathcal{V}(\bar{s})$ is defined through the probability mass function P_v, so that $\mathbb{P}(v_n = \bar{v}|s_n = \bar{s}) = P_v(\bar{v}, \bar{s})$. The degradation level is restricted to the finite set $s_n \in \{0, \ldots, 100\} = \mathcal{S}$: at $s_n = 0$ the boiler is perfectly clean and operates efficiently, whereas at $s_n = 100$ the boiler cannot operate (e.g., it is fully clogged). The dependence of the domain of $\mathcal{V}$ on s_n ensures that s_{n+1} does not leave $\mathcal{S}$, e.g. if $s_n = 99$, $\mathcal{V}(99) = \{0, 1\}$.

The degradation level is determined by a number of factors, and can be estimated as the ratio of power demanded (as a function of the desired temperature) to the power output of the boiler. These quantities are not known exactly, but rather may be measured by possibly noisy sensors. We model this uncertainty through an *observed* degradation level,

$$y_n = s_n + w_n, \quad (3)$$

[1]Optimal predictive maintenance is well motivated. Maintenance costs contribute between 15 and 60 percent of the cost of goods produced [17], and an efficient maintenance policy can reduce operational costs by 5 to 40 percent [18]. Although optimal maintenance is often treated as a simple case study in dynamic control texts [19], [20], there are few publications [21].

where the true degradation level s_n is corrupted by a noise term $w_n \in \mathcal{W}(s_n)$, where the noise is stochastic and takes values within the set $\mathcal{W}(s_n)$, again dependent on the actual degradation level. The probability that $w_n = \bar{w}$ given $s_n = \bar{s}$ is defined by the probability mass function P_w, namely $\mathbb{P}(w_n = \bar{w}|s_n = \bar{s}) = P_w(\bar{w}, \bar{s})$. Because we know that $s_n \in \{0, 100\}$, the observations are also restricted to the set $\mathcal{Y} = \{0, \ldots, 100\}$.

The control actions available at each time step are twofold: either to clean the boiler, which resets its degradation level, or to do nothing. In the first instance, the difference equation (2) for a single time step changes to

$$s_{n+1} = z_n, \tag{4}$$

with z_n a random variable taking values in $\mathcal{Z} = \{0, \ldots, 5\}$ with probabilities given by a distribution P_z.

We can therefore model the dynamical system (2), (3), and (4) as a POMDP with state space $\mathcal{S}$, observation space $\mathcal{Y}$, and control space $\mathcal{U} = \{\text{Clean}, \text{NoClean}\}$. The transition function T is derived from (2), (4), P_v, and P_z, so that

$$T(s'|s,u) = \begin{cases} P_v(s'-s, s), & \text{if } s'-s \in \mathcal{V}(s),\ u = \text{NoClean} \\ P_z(s'), & \text{if } s' \in \mathcal{Z},\ u = \text{Clean} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The observation function is derived from (3) and P_w as

$$Y(y|s) = \begin{cases} P_w(y-s, s), & \text{if } y-s \in \mathcal{W}(s) \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

We will assume that the initial distribution R is concentrated at a single point s^0, i.e. $R(s^0) = \mathbb{P}(s_0 = s^0) = 1$, although any well-defined probability mass function over $\mathcal{S}$ is acceptable.

C. Optimal Control of POMDPs

The optimal policy that minimises an expected sum of costs, as in (1), can be synthesised via dynamic programming, much like for an MDP, in which the state s_n is known completely [19]. A value function over the fully observed state, which represents the expected sum of costs accrued from time n to time N given the state at time n, is set and optimised backward-recursively for each time step n. The optimal policy for a POMDP is found by redefining the state space as something that is fully observable, and thus reformulating the POMDP as an MDP [19].

One way to redefine the POMDP as an MDP is to treat the information vector i_n as the fully observed state of the system. The information vector, however, increases in size with n, and can thus be difficult to store. A common alternative is to instead use a belief state, which is a sufficient statistic for the information vector, and therefore condenses the information stored in i_n without sacrificing the ability to construct optimal policies. For an additive objective function like (1), the belief state is a distribution that describes the probability of being in state s, given all past observations and actions [19], namely $b(s_n) = \mathbb{P}[s_n|u_0, \ldots, u_{n-1}, y_1, \ldots, y_n]$. By treating the belief state as the true state of the system, (1) can be equivalently solved by generating and recursively solving a value function over the belief state, namely

$$V_n^u(b) = \sum_s C(s,u)b(s) + \sum_y V_{n+1}^*\left(M_{y,u}b\right)\mathbb{P}(y|b,u), \tag{7}$$

initialised at N as $V_N^*(b) = \sum_s C(s,0)b(s)$, and taking $V_n^*(b) = \min_{u\in\mathcal{U}} V_n^u(b)$. The transition operator $M_{y,u}b$ provides the next belief state b_{n+1} given the current observation, action, and belief state according to a Bayesian update

$$\left(M_{y,u}b\right)(s') = \frac{Y(y|s',u)\sum_{s\in\mathcal{S}} T(s'|s,u)b(s)}{\mathbb{P}(y|b,u)}, \tag{8}$$

with the likelihood of the observation given by

$$\mathbb{P}(y|b,u) = \sum_{s\in\mathcal{S}} b(s) \sum_{s'\in\mathcal{S}} T(s'|s,u)Y(y|s',u). \tag{9}$$

An optimal policy π^* for the POMDP is then defined in terms of the belief state, and maps beliefs to actions: $\pi^* : \mathfrak{B} \to \mathcal{U}$ ($\mathfrak{B}$ denotes the space of all beliefs). The optimal policy π^* is obtained directly from the value function as $\pi_n^*(b) = \arg\min_{u\in\mathcal{U}} V_n^u(b)$.

The value function in (7) for a POMDP is often represented in an alternate form that makes it conceptually easier to solve. If we pose the problem as a maximization (i.e. maximize the sum of costs multiplied by -1), then for a finite horizon $N < \infty$, the value function at each time n is piecewise-linear and convex in the belief state [22], and thus can be expressed as

$$V_n^*(b) = \max_{\alpha_n\in\Gamma_n} \sum_s \alpha_n(s)b(s). \tag{10}$$

The functions $\alpha_n \in \mathbb{R}^{|\mathcal{S}|}$, or "$\alpha$-vectors", represent a bounded portion of the value function, and characterise the current value of being in each state $s \in \mathcal{S}$, given a specific action u is taken, plus the expected sum of future rewards assuming that all subsequent actions are chosen optimally. Because each α-vector is associated with a specific action, by picking the α-vector maximising $\sum_s \alpha_n(s)b(s)$ we also define the optimal policy for belief b at time n. The collection of α-vectors needed to exactly represent the value function V_n at time n is finite and represented by a set denoted as Γ_n.

The α-vectors at time n are computed recursively from the α-vectors known at time $n+1$. For each action, we access one of $|\mathcal{Y}|$ observations, and for each observation there is a subsequent α-vector defined at time $n+1$, resulting in $|\mathcal{U}||\Gamma_{n+1}|^{|\mathcal{Y}|}$ α-vectors at time n. Rather than computing and storing an exponentially growing set of α-vectors, one option is to use point-based value iteration (PBVI), a sampling-based method used to approximate value functions [23].

There are many variants of PBVI around a basic algorithm. A finite subset $B \subset \mathfrak{B}$ is generated through some form of sampling, and for each $b \in B$, the value function is estimated by producing a single α-vector for each sampled belief state at each iteration. Therefore, if $B = \{b^0, b^1, \ldots, b^m\}$, then $\tilde{\Gamma}_n = \{\alpha_n^0, \alpha_n^1, \ldots, \alpha_n^m\}$ for all n, and the collection of α-vectors does not increase in size at each iteration. For a finite horizon problem, we distinguish the belief state at different times, so that $B = \bigcup_{n=0}^N B_n$, with $B_n = \{b_n^1, \ldots, b_n^{m_n}\}$ and $B_0 = \{R\}$, the initial distribution of the POMDP of interest.

For an additive objective function, the α-vectors are computed as in [16]. First, for each belief state b^k and at each time step n, a set of intermediate α-vectors $\alpha^j_{n,y,u}$ are computed recursively from $\tilde{\Gamma}_{n+1}$

$$\alpha^j_{n,y,u}(s) = C(s,u) + \sum_{s'} \alpha^j_{n+1}(s')Y(y|s')T(s'|s,u) \quad (11)$$

for $\alpha^j_{n+1} \in \tilde{\Gamma}_{n+1}$. Then, for each b^k we define $j^*(k) = \arg\max_j \sum_s \alpha^j_{n,y,u}(s)b^k(s)$. The final α-vectors associated with a single control input and belief state are given as

$$\alpha^k_{n,u}(s) = \sum_y \alpha^{j^*(k)}_{n,y,u}(s) \quad (12)$$

and $\alpha^k_n = \alpha^k_{n,u^*}(s)$, with $u^* = \arg\max_{u\in\mathcal{U}} \sum_s \alpha^k_{n,u}(s)b^k(s)$, is added to $\tilde{\Gamma}_n$.

An α-vector α^j_n corresponding to b^j will likely apply to all belief points in a region around b^j (i.e. for any b in a neighborhood of b^j the same action will likely be optimal). Hence the value at some b not necessarily in B can be approximated by $V^*_n(b) \approx \max_{\alpha^i_n\in\tilde{\Gamma}_n} \sum_s \alpha^i_n(s)b(s)$ as in (10) but with the restricted set $\tilde{\Gamma}_n \subset \Gamma_n$. The PBVI algorithm then consists of selecting a set of belief points B_n, and computing α^k_n for each $b^k_n \in B_n$.

The convexity of the value function guarantees that by storing only a subset $\tilde{\Gamma}_n$ of α-vectors, the approximation provides a lower bound to the maximum expected cost at any belief state (or an upper bound to the minimum), i.e. $\max_{\alpha^i_n\in\tilde{\Gamma}_n} \sum_s \alpha^i_n(s)b(s) \le \max_{\alpha^i_n\in\Gamma_n} \sum_s \alpha^i_n(s)b(s)$ for any $b \in \mathfrak{B}$.

D. Safety Objectives for POMDPs

Rather than only minimising an expected sum of costs as in (1), we consider the specific task of generating safety-preserving controllers (where safety is preserved with maximal probability), with the additional objective of minimising the associated costs. This is a multi-objective goal with heterogeneous components (safety and performance). The techniques presented can, however, be extended to more than two separate objectives, and also to other complex specifications beyond safety (reach-avoid, reachability, etc.).

Specifically, consider a safe or desired subset of the state space, denoted $K \subset \mathcal{S}$, in which we would like the state s_n to remain over a finite time horizon $n = 0, \ldots, N$. We would like to design a control policy that maximises the probability that $s_n \in K$ for $n = 0, \ldots, N$ and to determine the value of that maximal probability – this task is known as "quantitative verification." In other words, we would like to find

$$p^N_{\text{safe}}(\pi, R; K) = \mathbb{P}^\pi\left[s_0 \in K, \ldots, s_N \in K | R\right], \quad (13)$$

$$\pi^*_{\text{safe}} = \arg\max_{\pi\in\Pi} p^N_{\text{safe}}(\pi, R; K). \quad (14)$$

We can express (13) using standard stochastic optimal control notation (i.e. the cost objective is an expected value of a function of one-step costs) by recalling that for a random variable s and set K, the probability that $s \in K$ is equal to the expected value of the indicator function over that event, $\mathbb{E}[\mathbf{1}_K(s)]$ [1]. Hence, (13) is equal to

$$p^N_{\text{safe}}(\pi, R; K) = \mathbb{E}^\pi\left[\prod_{n=0}^N \mathbf{1}_K(s_n)\middle| R\right].$$

Maximizing the probability of safety is therefore equivalent to maximizing a *multiplicative* objective function (rather than additive, as discussed in Section II-C). The belief state $b(s_n) = \mathbb{P}[s_n \mid i_n]$ is *no longer* sufficient when the objective is multiplicative, because the costs accrued at previous time steps must also be considered, as they affect the current possible cost (i.e. if any previous state left the safe set, the system can not in subsequent time steps be considered safe). Two options to construct an equivalent MDP and value function over a fully observable state to compute (14) are 1) To derive a sufficient statistic and to redefine the belief state directly over the multiplicative objective function, as in [3], or 2) To reformulate the POMDP so that (13) can be framed as an additive objective function, as in [24]. While both approaches are equivalent, we will present the second option, because to apply the lexicographic approach for multiple objectives, we need each cost function to have the same format. Since all other cost objectives we consider in this work are additive (like minimizing the sum of costs accrued at each times step), we will pose safety as an additive cost objective.

To express (13) as an expected value over a sum of costs, an additional binary state variable q_n is introduced (which is not observed),

$$q_n = \prod_{i=0}^{n-1} \mathbf{1}_K(s_i). \quad (15)$$

The variable q_n keeps track of whether the state of the system has remained within K up to the previous time step, thus keeping track of the previous "costs" incurred (where the cost is simply equal to one or zero, depending on whether the system has remained safe or not).

The new state of the system is $\bar{s} = (s, q) \in \mathcal{S} \times \{0, 1\} = \bar{\mathcal{S}}$, and the safety objective can be rewritten as

$$\max_{\pi\in\Pi} \mathbb{E}^\pi\left[\prod_{n=0}^N \mathbf{1}_K(s_n)\middle| R\right] = \max_{\pi\in\Pi} \mathbb{E}^\pi\left[\mathbf{1}_{K\times\{1\}}(s_N, q_N)\middle| R\right]. \quad (16)$$

The cost function from $n = 0, \ldots, N-1$ is $C(\bar{s}_n, u_n) = 0$, and at time N, $C(\bar{s}_N, u_N) = \mathbf{1}_{K\times\{1\}}(s_N, q_N)$, and so (16) is an additive objective function with zero costs accrued until the final time step.

The transition probability function T is replaced by $\bar{T} : \bar{\mathcal{S}} \times \bar{\mathcal{S}} \times \mathcal{U} \to [0, 1]$, with

$$\bar{T}(s', q'|s, q, u) = \begin{cases} T(s'|s, u), & \text{if } q = q' = 0 \\ 0, & \text{if } q = 0,\ q' = 1 \\ \mathbf{1}_{\mathcal{S}\setminus K}(s)T(s'|s, u), & \text{if } q = 1,\ q' = 0 \\ \mathbf{1}_K(s)T(s'|s, u), & \text{if } q = q' = 1 \end{cases}. \quad (17)$$

We also transform $Y(y|s)$ to $\bar{Y}(y|\bar{s})$, such that $\bar{Y}(y|s, q) = Y(y|s)$, which is independent of q, and similarly transform $R(s)$ to $\bar{R}(s, q) = R(s)$.

Therefore, the problem of maximizing the probability of safety for a POMDP $\mathcal{J}$ can be posed equivalently as an additive cost problem over a different POMDP $\bar{\mathcal{J}} = (\bar{\mathcal{S}}, \mathcal{Y}, \mathcal{U}, \bar{T}, \bar{Y}, \bar{R})$. The belief state is now the conditional probability distribution of the current state of the system conditioned on all past observations and control inputs, only now the current state is $\bar{s}$ rather than just s: $\bar{b}(\bar{s}_n) = \mathbb{P}[s_n, q_n | i_n]$.

III. Problem Formulation

We would like to consider the problem of maximising the probability that the state of a POMDP remains safe over a finite time horizon, while simultaneously minimising an expected sum of costs. It may be the case that the control policy π^* from (14) that produces the highest probability that the system remains safe is simply not feasible to implement from a cost perspective. It may also be the case that a small sacrifice in the optimal safety level (or probability) of the system leads to a significant reduction in costs. Depending on the requirements on safety-criticality of the system at hand, we can define a tolerance η, which indicates how much we are willing to lower the probability of safety from the optimal one, in order to attempt to reduce the expected cost by implementing a (sub-)optimal policy.

We will denote by $\tilde{\Pi}$ the set of control policies that produce a probability of safety within the desired tolerance (no more than η below the maximum), namely

$$\tilde{\Pi} = \{\pi : p^N_{\text{safe}}(\pi^*_{\text{safe}}, R; K) - p^N_{\text{safe}}(\pi, R; K) \leq \eta\}. \quad (18)$$

We would then like to find the policy $\tilde{\pi}^* \in \tilde{\Pi}$ that minimises the expected sum of costs:

$$\tilde{\pi}^* = \arg\min_{\pi \in \tilde{\Pi}} \mathbb{E}^{\pi} \left[\sum_{n=0}^{N} C(s_n, u_n) \middle| R \right]. \quad (19)$$

We therefore have a multi-objective optimisation problem of a hierarchical nature, where one objective takes priority over the other, and whose cost functions are inherently different.

Problem 1: Given a POMDP $\mathcal{J}$, a safe set $K \subset \mathcal{S}$, a cost function $C(s, u)$, a time horizon $N < \infty$, and a tolerance η, we would like to synthesize a control policy $\tilde{\pi}^*$ that

1) Guarantees that the probability that the system state s_n remains in K for all $n = 0, \ldots, N$ (13) is no less than η below the maximum possible safety probability, i.e.

$$p^N_{\text{safe}}(\pi^*_{\text{safe}}, R; K) - p^N_{\text{safe}}(\tilde{\pi}^*, R; K) \leq \eta, \quad (20)$$

2) Guarantees that the expected cost using policy $\tilde{\pi}^*$ is minimal over all policies that produce a safety probability within η of the optimal, i.e. $\tilde{\pi}^*$ is the solution to (19).

1) Case Study Continued: Returning to the boiler maintenance problem, we have two main objectives. The first is to ensure that the boiler degradation level does not reach 80, at which point the boiler is essentially broken and would result in excessive maintenance costs and possible loss of heat over an extended period of time. To avoid this unsafe outcome, the preventive maintenance action "Clean" may be taken, which incurs a cost but is less costly than if the boiler were to break.

In addition, the higher the degradation level, the less efficient the boiler is, and the more expensive it is (the more fuel required) for the boiler to maintain a set temperature. It is therefore preferable to do preventive maintenance, and to clean the boiler before the degradation level grows too high both to avoid a breakdown, and also to reduce fuel costs.

The total cost at each time step, assuming the degradation level has not reached 80, is given by the function

$$C(s, u) = 0.05\, s + 100\, \mathbf{1}_{\text{Clean}}(u). \quad (21)$$

The first term captures efficiency costs associated with the degradation level, and the second gives the cost of cleaning the boiler (100 Euros) if $u = \text{Clean}$ is selected.

Rather than assign a cost to the event that the degradation level reaches 80, we treat the set $K^c = \{80, \ldots, 100\}$ as an unsafe set that we would like to avoid, and we wish to find a cleaning policy (at what point the boiler should be cleaned) that maximises the probability that the unsafe set is not reached. We would further like to *know* this probability, so that we can provide guarantees on the safety of the boiler under the policy we propose.

If we were to only consider safety, an intuitive optimally safe policy would be to clean the boiler all the time, regardless of the degradation level. From a cost standpoint, however, this is not practical, and we should also factor in the fuel costs, and costs associated with cleaning, when designing a safe policy.

IV. Lexicographic Optimal Control

We consider now multiple objective functions to be maximised concurrently: we shall employ the same belief state, which by definition does not depend on the cost function so long as the costs incurred are additive at each time step, as the input to multiple value functions, one for each cost objective. We can then set up a POMDP with lexicographic optimization criteria, introduced in [15], with k cost functions $\{C_1, \ldots, C_k\}$ and with tolerance parameters $\{\eta_1, \ldots, \eta_{k-1}\}$ associated with the first $k-1$ cost functions. The cost functions are ordered so that C_1 is the most important to optimise, and C_k the least important. Each cost function C_i is optimised using the value function $V_{n,i}$, defined as in (7) with C replaced by C_i.

The idea behind the lexicographic approach is to progressively limit the set of control inputs available to optimise the value functions $V_{n,i}(b)$ at each time step n, and for each belief state, according to the preference ordering. The restricted set of control inputs available to value function $V_{n,i+1}(b)$ for a specific belief state b, denoted $\mathcal{U}_{n,i}(b)$, is chosen such that the value function $V^u_{n,i}(b)$ (evaluated at a specific $u \in \mathcal{U}_{n,i-1}$) is within $\bar{\eta}_i$, the *one step* slack allowance (as opposed to η_i, which is the total slack allowance over the entire time horizon), of $V^*_{n,i}(b)$, and thus ultimately ensures that the policy $\tilde{\pi}^*$ generated by selecting the optimal control input $u^*_n \in \mathcal{U}_{n,k-1}(b)$ for any belief b at any time n, is within $\tilde{\Pi}$.

The actual procedure is as follows. Starting at the final time step N, we construct $V^u_{N,i}(b) = \sum_s C_i(s, u) b(s)$ for all $i = 1, \ldots, k$. For each $b \in \mathfrak{B}$, we then construct the set of permissible control inputs $\mathcal{U}_{N,i}(b)$ for all $i = 1, \dots, k-1$, defined as follows:

$$\mathcal{U}_{N,i}(b) = \left\{ u \in \mathcal{U}_{N,i-1} : \max_{\bar{u} \in \mathcal{U}_{N,i-1}} V^{\bar{u}}_{N,i}(b) - V^{u}_{N,i}(b) \leq \bar{\eta}_i \right\}. \tag{22}$$

How to choose $\bar{\eta}_i$ under various conditions will be described subsequently. The set $\mathcal{U}_{N,0}$ is simply $\mathcal{U}$.

Once $\mathcal{U}_{N,k-1}(b)$ is computed, we find $u^* = \arg\max_{u \in \mathcal{U}_{N,k-1}} V^u_{N,k}(b)$. We then set $V^*_{N,i}(b) = \sum_s C_i(s, u^*) b(s)$. The process is repeated for all $b \in \mathfrak{B}$, and for all time steps $n = N-1, \dots, 0$, using the value function in (7) with each cost function C_i, and with the restricted control inputs $\mathcal{U}_{n,i}(b)$.

The α-vector representation of the value function associated with a POMDP allows us to alternatively characterise the lexicographic preferences through sets of α-vectors, rather than the sets of control inputs $\mathcal{U}_{n,i}(b)$. Similar to the sets $\mathcal{U}_{n,i}(b)$, we generate sequential sets $\bar{\Gamma}_{n,i} \subset \Gamma_{n,i}$, with the difference being that the sets $\bar{\Gamma}_{n,i}$ are not generated separately for each belief state. For time N, we set $\bar{\Gamma}_{N,1} = \Gamma_{N,1}$, and for $n < N$, $\bar{\Gamma}_{n,1} = \hat{\Gamma}_{n,1}$, where $\hat{\Gamma}_{n,1}$ is the full set of α-vectors (for all possible control inputs) computed recursively from $\bar{\Gamma}_{n+1,1}$. For $i = 2, \dots, k$,

$$\bar{\Gamma}_{n,i} = \left\{ \alpha \in \hat{\Gamma}_{n,i} : \exists\, b \in \mathfrak{B} \text{ s.t. } \max_{\alpha' \in \bar{\Gamma}_{n,i-1}} \alpha' \cdot b - \alpha'' \cdot b \leq \bar{\eta}_i \right.$$
$$\left. \text{with } \alpha'' \in \bar{\Gamma}_{n,i-1}, \text{ and } u_\alpha = u_{\alpha''} \right\}. \tag{23}$$

The notation u_α refers to the control input associated with vector α. In other words, the control inputs allowed when generating the α-vectors for $\bar{\Gamma}_{n,i}$ are restricted to those associated with α-vectors in $\bar{\Gamma}_{n,i-1}$ such that there exists some $b \in \mathfrak{B}$ for which the inner product between the α-vector and b is within $\bar{\eta}_i$ of the optimal. We no longer compute and store all α-vectors for each value function $V_{n,i}$, but rather prune according to whether they meet the tolerance specification for the previous cost function.

The notation can be quite cumbersome: we have included a summary of the relevant notation and its meaning in Table I.

Notation	Meaning
C_i	Cost function i
$V_{n,i}$	Value function at time n for C_i
$V^u_{n,i}$	$V_{n,i}$ evaluated at input u
$V^*_{n,i}$	$\max_{u \in \mathcal{U}} V^u_{n,i}$
η_i	Total slack tolerance
$\bar{\eta}_i$	One step slack tolerance
$\mathcal{U}_{n,i}(b)$	Restricted set of control inputs available to $V_{n,i+1}(b)$
$\hat{\Gamma}_{n,i}$	$\subseteq \Gamma_{n,i}$, subset of all α-vectors that approximate $V_{n,i}$
$\bar{\Gamma}_{n,i}$	$\subseteq \hat{\Gamma}_{n,i}$ set that satisfy tolerance specification for $V_{n,i-1}$
$\tilde{\bar{\Gamma}}_{n,i}$	$\subseteq \bar{\Gamma}_{n,i}$ to approximate $V_{n,i}$ using PBVI

TABLE I: List of notation used for the lexicographic optimization procedure.

A. Application to a Prioritised Safety Objective

The lexicographic optimal control approach can be used when the priority is a safety objective in (13) by using the additive objective formulation that relies on including the binary variable q_n as part of the state, as described in Section II-D. This formulation allows us to use the same belief states for the value functions associated with both the safety and cost objectives. The value functions for the safety objective are

$$\begin{aligned} V^u_{N,1}(\bar{b}) &= \sum_{s,q} \mathbf{1}_{K \times \{1\}}(s,q) \bar{b}(s,q), \\ V^u_{n,1}(\bar{b}) &= \sum_{s,q} V^*_{n+1,1}(\bar{M}_{y,u}\bar{b}) \mathbb{P}(y|\bar{b}, u), \end{aligned} \tag{24}$$

with $\bar{M}$ and $\mathbb{P}(y|\bar{b}, u)$ defined as in (8) and (9), only with T replaced by $\bar{T}$, Y replaced by $\bar{Y}$, and summations over $\bar{s}$. The value function for minimizing the cost function C is

$$\begin{aligned} V^u_{N,2}(\bar{b}) &= \sum_{s,q} C(s,u) \bar{b}(s,q), \\ V^u_{n,2}(\bar{b}) &= \sum_{s,q} C(s,u) \bar{b}(s,q) + \sum_{y} V^*_{n+1,2}(\bar{M}_{y,u}\bar{b}) \mathbb{P}(y|\bar{b}, u). \end{aligned} \tag{25}$$

Notice that the variable q_n is essentially ignored in the value function $V_{n,2}$, and $\bar{b}(\bar{s})$ is reduced to $b(s)$ by treating $b(s)$ as the marginal distribution, namely $b(s) = \sum_q \bar{b}(s,q)$.

Problem 1 can then be solved in theory using the above belief state and value functions, and by applying the lexicographic approach described above with $k = 2$ separate cost functions, and a single tolerance parameter η (and also a single one-step tolerance parameter $\bar{\eta}$). As seen with a single objective function, however, (24) and (25) cannot be solved exactly, but instead approximated using a modification of PBVI.

B. Approximate Solution using PBVI

In order to use PBVI in a lexicographic setting with both a safety and a cost objective, we need to compute two sets of α-vectors, one for the safety objective, and one for the cost, which we denote $\alpha^k_{n,1}$ and $\alpha^k_{n,2}$, respectively. The doubly restricted sets of α-vectors, denoted $\tilde{\bar{\Gamma}}_{n,i}$ (restricted because we compute less α-vectors for the PBVI algorithm, and because we store different α-vectors depending on the tolerance parameter η) are generated according to the standard PBVI algorithm, with two modifications.

The first modification is in computing the index $j^*(k)$. If we were to compute the exact solution, the α-vectors for *all possible* combinations of observations and control inputs would be generated, independently of the belief state, and therefore the selection of the optimal index $j^*(k)$ is not necessary. However, when applying PBVI, we must be careful to ensure that the index $j^*(k)$ computed for $\alpha^{j^*(k)}_{n,y,u,1}$ and $\alpha^{j^*(k)}_{n,y,u,2}$ is the same, i.e. the same α-vector $\alpha^{j^*(k)}_{n+1}$, and hence the same optimal policy at subsequent time steps, is associated with both $\alpha^k_{n,1}$ and $\alpha^k_{n,2}$. Otherwise, different optimal policies would be associated with the safety and cost objectives, implying the selection of different control inputs for the same belief state at time $n+1$ depending on which objective is considered. Since the idea

Algorithm 1 LexicOpt

Input: $\{\alpha_1^j\}_{j=1}^M$, $\{\alpha_2^j\}_{j=1}^M, b, \bar{\eta}$
Output: j^{**}, optimal index associated with α_1, α_2

```
 1: $j^{**} \leftarrow \arg\max_j \sum_{s\in\mathcal{S}} \alpha_1^j(s)b(s)$
 2: maxSafe $\leftarrow \sum_{s\in\mathcal{S}} \alpha_1^{j^{**}}(s)b(s)$
 3: minCost $\leftarrow \sum_{s\in\mathcal{S}} \alpha_2^{j^{**}}(s)b(s)$
 4: for $j = 1,\dots,M$ do
 5:     newSafe $\leftarrow \sum_{s\in\mathcal{S}} \alpha_1^j(s)b(s)$
 6:     newCost $\leftarrow \sum_{s\in\mathcal{S}} \alpha_2^j(s)b(s)$
 7:     if maxSafe $-$ newSafe $\le \bar{\eta}$ then
 8:         if newCost $<$ minCost then
 9:             $j^{**} \leftarrow j$
10:             minCost $\leftarrow \sum_{\bar{s}\in\mathcal{S}} \alpha_2^{\bar{j}}(s)b(s)$
11:         end if
12:     end if
13: end for
```

is to select a *single* control input to simultaneously optimize both objectives, this cannot be allowed.

Further, the term $\sum_{s'} \alpha_{n+1}^j(s')Y(y|s')T(s'|s,u)$ in (11) is proportional to $\sum_{s'} \alpha_{n+1}^j(s')b_{n+1}(s')$. Hence when $j^*(k)$ is selected, we are actually optimizing the value function at time $n+1$. The b_{n+1} produced by (11) may not, however, be one of the elements in B_{n+1}, and so the exact optimal α_{n+1}^j likely has not been computed. The implication when performing the lexicographic optimization is that the (sub)optimal α_{n+1}^j designed to maximize the safety objective to within $\bar{\eta}$ of the optimal, and minimize the cost objective, likely has also not been computed. Therefore the lexicographic optimization must be redone for the belief state $\bar{b}_{n+1}$ generated by $\alpha_{n,y,u,i}^j$, and the tolerance preference $\bar{\eta}$ must once again be enforced. The modified index, denoted $j^{**}(k)$, is computed according to Algorithm 1, with inputs $\{\alpha_{n,y,u,1}^j\}_j$, $\{\alpha_{n,y,u,2}^j\}_j$, $\bar{b}^k$, and $\bar{\mathcal{S}}$. Although PBVI is mentioned as an approximation algorithm for the lexicographic optimization of POMDPs in [15], they fail to mention this subtle difference from the standard, single objective PBVI algorithm.

The α-vectors $\alpha_{n,u,1}^k$ and $\alpha_{n,u,2}^k$ (defined for a specific control input u and for each $\bar{b}^k \in \bar{B}_n$, $n = 0,\dots,N$) are given by

$$\begin{aligned}\alpha_{N,u,1}^k(\bar{s}) &= \mathbf{1}_{K\times\{1\}}(\bar{s}),\\ \alpha_{n,u,1}^k(\bar{s}) &= \sum_{y\in\mathcal{Y}}\sum_{\bar{s}'\in\bar{\mathcal{S}}} \alpha_{n+1,1}^{j^{**}(k)}(\bar{s}')\bar{Y}(y|\bar{s}')\bar{T}(\bar{s}'|\bar{s},u).\end{aligned} \tag{26}$$

$$\begin{aligned}\alpha_{N,u,2}^k(\bar{s}) &= C(s,u),\\ \alpha_{n,u,2}^k(\bar{s}) &= C(s,u) + \sum_{y\in\mathcal{Y}}\sum_{\bar{s}'\in\bar{\mathcal{S}}} \alpha_{n+1,2}^{j^{**}(k)}(\bar{s}')\bar{Y}(y|\bar{s}')\bar{T}(\bar{s}'|\bar{s},u).\end{aligned} \tag{27}$$

Note that $\alpha_{n,u,1}^k$ and $\alpha_{n,u,2}^k$ are functions of $\alpha_{n+1,1}^j$ and $\alpha_{n+1,2}^j$, respectively, such that the index u at time $n+1$ is dropped. This is because the dependence on u is captured within the index j, according to the next modification to the PBVI algorithm.

The second modification is the same as for the exact lexicographic solution described in Section II-C. Once the complete sets of α-vectors $\alpha_{n,u,i}^k$ are computed according to (26) and (27), the sets $\tilde{\bar{\Gamma}}_{n,i}$ are generated by first constructing the set $\mathcal{U}_n(\bar{b}^k)$, which can be expressed in terms of the α-vectors as

$$\mathcal{U}_n(\bar{b}^k) = \left\{u\in\mathcal{U} : \max_{\bar{u}\in\mathcal{U}} \sum_{\bar{s}} \alpha_{n,\bar{u},1}^k(\bar{s})\bar{b}^k(\bar{s}) - \sum_{\bar{s}} \alpha_{n,u,1}^k(\bar{s})\bar{b}^k(\bar{s}) \le \bar{\eta}\right\}. \tag{28}$$

The optimal control input u^* is chosen as

$$u^* = \arg\min_{u\in\mathcal{U}_n(\bar{b}^k)} \sum_{\bar{s}} \alpha_{n,u,2}^k(\bar{s})\bar{b}^k(\bar{s}), \tag{29}$$

and $\alpha_{n,1}^k = \alpha_{n,u^*,1}^k$ and $\alpha_{n,2}^k = \alpha_{n,u^*,2}^k$ are added to $\tilde{\bar{\Gamma}}_{n,1}$ and $\tilde{\bar{\Gamma}}_{n,2}$, respectively. Algorithm 2 summarizes the entire procedure for solving Problem 1 using PBVI and a lexicographic approach.

Note that Algorithm 2 returns the sets $\{\tilde{\bar{\Gamma}}_{n,1}\}_{n=0}^N$ and $\{\tilde{\bar{\Gamma}}_{n,2}\}_{n=0}^N$, which in turn provide the multi-objective policy $\tilde{\pi}^*$. The optimal control input at time n, for any belief state $\bar{b} \in \bar{\mathfrak{B}}$, is found again according to Algorithm 1, with inputs $\tilde{\bar{\Gamma}}_{n,1}$, $\tilde{\bar{\Gamma}}_{n,2}$, and $\bar{b}$. Using the returned index i^*, u^* is equal to the control input associated with both $\alpha_{n,1}^{i^*}$ and $\alpha_{n,2}^{i^*}$, which is guaranteed to be the same based on Algorithm 2.

The probability that the system remains safe, if policy $\tilde{\pi}^*$ is implemented, and the expected cost associated with the policy, are bounded by

$$\begin{aligned} p_{\text{safe}}^N(\tilde{\pi}^*, R; K) &\ge \sum_{\bar{s}} \alpha_{0,1}^1(\bar{s})\bar{R}(\bar{s})\\ \mathbb{E}^{\tilde{\pi}^*}\left[\sum_{n=0}^N C(s_n, \tilde{\pi}_n^*(\bar{b}_n))\middle| R\right] &\le \sum_{\bar{s}} \alpha_{0,2}^1(\bar{s})\bar{R}(\bar{s}),\end{aligned} \tag{30}$$

since $\bar{b}^1 = \bar{R}$.

C. Bound on the One-Step Tolerance

We next discuss how to select the one step tolerance $\bar{\eta}$ to ensure that the approximate safety probability returned by PBVI is within tolerance η of the exact optimal safety probability, $p_{\text{safe}}^N(\pi_{\text{safe}}^*, R; K)$. The proof is similar to that in [15], although without the modifications to the PBVI algorithm discussed in the previous section, the claims in [15] are incorrect. We first derive the one-step tolerance needed to guarantee an overall tolerance η, if we were able to solve the lexicographic optimization exactly (without PBVI).

Proposition 1: For time horizon N, initial distribution R, and total desired tolerance η, setting the one step tolerance $\bar{\eta}$ to $\bar{\eta} = \frac{\eta}{N}$ ensures that the safety probability found by exactly solving the lexicographic POMDP optimization, denoted $V_{0,1}^{\eta}(R)$ satisfies $p_{\text{safe}}^N(\pi^*, R; K) - V_{0,1}^{\eta}(R) \le \eta$.

As an extension, in order to ensure that the selected tolerance is not exceeded when using PBVI, we must also take

Algorithm 2 LexicoPBVI

Input: $\{B_n\}_{n=0}^N$, K, $C(\cdot,\cdot)$, N, $\mathcal{J}$, $\bar{\eta}$
Output: $\{\tilde{\bar{\Gamma}}_{n,1}\}_{n=0}^N$, $\{\tilde{\bar{\Gamma}}_{n,2}\}_{n=0}^N$

```
1: $\tilde{\bar{\Gamma}}_{N,1} = \emptyset$, $\tilde{\bar{\Gamma}}_{N,2} = \emptyset$
2: $\alpha_{N,1}(\cdot) = \mathbf{1}_K(\cdot)$, $\tilde{\bar{\Gamma}}_{N,1} = \tilde{\bar{\Gamma}}_{N,1} \bigcup \{\alpha_{N,1}\}$
3: for all $b \in B_N$ do
4:     $u^* = \arg\min_{u \in \mathcal{U}} \sum_{s \in \mathcal{S}} C(s,u)b(s)$
5:     $\alpha_{N,2}(\cdot) = C(\cdot, u^*)$, $\tilde{\bar{\Gamma}}_{N,2} = \tilde{\bar{\Gamma}}_{N,2} \bigcup \{\alpha_{N,2}\}$
6: end for
7: for $n = N-1, \ldots, 0$ do
8:     $\tilde{\bar{\Gamma}}_{n,1} = \emptyset$, $\tilde{\bar{\Gamma}}_{n,2} = \emptyset$
9:     for all $b^k \in B_n$ do
10:        for all $u \in \mathcal{U}$ do
11:            $\alpha^k_{n,u,1} = \mathbf{0}$, $\alpha^k_{n,u,2} = \mathbf{0}$
12:            for all $y \in \mathcal{Y}$ do
13:                for $j = 1, \ldots, |\tilde{\bar{\Gamma}}_{n+1,1}|$ do
14:                    Compute $\alpha^j_{n,y,u,1}$, $\alpha^j_{n,y,u,2}$ according to (26)
                       and (27), respectively
15:                end for
16:                $j^{**}(k) = \text{LexicOpt}(\{\alpha^j_{n,y,u,1}\}, \{\alpha^j_{n,y,u,2}\}, b^k, \bar{\eta})$
17:                $\alpha^k_{n,u,1} = \alpha^k_{n,u,1} + \alpha^{j^{**}(k)}_{n,y,u,1}$
18:                $\alpha^k_{n,u,2} = \alpha^k_{n,u,2} + \alpha^{j^{**}(k)}_{n,y,u,2}$
19:            end for
20:        end for
21:        $u^* = \text{LexicOpt}(\{\alpha^k_{n,u,1}\}_u, \{\alpha^k_{n,u,2}\}_u, b^k, \bar{\eta})$
22:        $\tilde{\bar{\Gamma}}_{n,1} = \tilde{\bar{\Gamma}}_{n,1} \bigcup \{\alpha^k_{n,u^*,1}\}$, $\tilde{\bar{\Gamma}}_{n,2} = \tilde{\bar{\Gamma}}_{n,2} \bigcup \{\alpha^k_{n,u^*,2}\}$
23:    end for
24: end for
```

into account the error introduced in the PBVI algorithm when maximising the safety objective. This error is given in [25] and is a straightforward extension of the standard PBVI error presented in [16]. First, we employ parameter δ^B to denote how densely the belief space $\mathfrak{B}$ has been sampled to produce B, defined as the maximum Hausdorff distance with respect to n between B_n and $\mathfrak{B}_n$:

$$\delta^B = \max_n \left\{ \max\{ \sup_{\tilde{b}_n \in B_n} \inf_{b_n \in \mathfrak{B}_n} \|\tilde{b}_n - \bar{b}_n\|_1, \sup_{b_n \in \mathfrak{B}_n} \inf_{\tilde{b}_n \in B_n} \|\tilde{b}_n - b_n\|_1 \right\}. \quad (31)$$

Denoting V_n^B as the optimal value function for a single objective at time n computed using PBVI, and V_n^* as the exact optimal value function, then

$$V_n^*(b) - V_n^B(b) \le (N-n)\delta^B \quad (32)$$

for all $b \in \mathfrak{B}$, and for all $n = 0, \ldots, N$. We obtain:

Proposition 2: For time horizon N, initial distribution R, and total desired tolerance η, if $\frac{\eta}{N} - \delta^B \ge 0$, setting the one step tolerance $\bar{\eta}$ to $\bar{\eta} = \frac{\eta}{N} - \delta^B$ ensures that the safety probability found by solving the lexicographic POMDP optimisation using PBVI, denoted $V_{0,1}^{\eta,B}(R)$ satisfies $p^N_{\text{safe}}(\pi^*, K; R) - V_{0,1}^{\eta,B}(R) \le \eta$.

Proposition 2 guarantees that the safety probability returned using Algorithm 2 will be within η of $p^N_{\text{safe}}(\pi^*, K; R)$, as long as the error $N\delta^B$ incurred as a result solely from the PBVI algorithm is not too large (i.e. does not exceed η). Fortunately, the PBVI error is tunable, and δ^B decreases to zero as the sampled set B approaches $\mathfrak{B}$. Unfortunately, the more points $b \in B$ that are generated, the more computation time is required for Algorithm 2, so there is an inherent trade-off between the accuracy of the PBVI algorithm and hence our ability to satisfy the tolerance specification η, and the amount of time required to compute the optimal policy for both the safety and cost objectives.

V. Extension to Hybrid Systems

By definition, cyber-physical systems typically comprise processes evolving in a continuous space (the physical system), in combination with processes evolving in a discrete fashion (the "cyber," or computational component embedded in the physical process). They are therefore often modeled within a hybrid system framework, with a state space $\mathcal{S} = \mathcal{X} \times \mathcal{Q}$, $\mathcal{X} \subseteq \mathbb{R}^n$, and $\mathcal{Q} = \{q_1, \ldots, q_l\}$ a finite set of modes. A partially observable hybrid system may also have a combination of continuous and discrete observations, so that $\mathcal{Y} = \mathcal{Y}^x \times \mathcal{Y}^q$, $\mathcal{Y}^x \subset \mathbb{R}^m$ and $\mathcal{Y}^q = \{y_1^q, \ldots, y_p^q\}$.

1) Case Study: Parsimonious Temperature Regulation: As an example of a CPS, consider the task of controlling a boiler to switch on or off to heat a single room in a building. The safe temperature range is between 17.5 and 22 degrees Celsius, and the safety objective is to construct a control policy to maximize the probability that the temperature in the room does not leave this range. Simultaneously we would like to limit the amount of time that the heater is on, to reduce operational costs, i.e. we want to minimize the cost function $\sum_{n=0}^N C(u_n)$, with $C(0) = 0$ and $C(1) = 1$.

The temperature evolution is approximated by an affine stochastic difference equation, as presented in [1]:

$$x_{n+1} = (1-a)x_n + bu_n + ax_a + v_n, \quad (33)$$

where x_n is the temperature at time step n, x_a is the ambient temperature (assumed constant), a and b are constants representing the heat loss rate to the external environment and the rate of heat supplied by the heater respectively, and $\{v_n\}$ is a sequence of i.i.d. Gaussian random variables with zero mean and variance ν^2 representing stochastic disturbances to the temperature.

While the temperature evolution is a continuous process, and the state takes values in $\mathcal{X} = \mathbb{R}$, the control input is a discrete signal $u \in \mathcal{U} = \{0, 1\}$ telling the boiler to switch on or off; 0 indicates the boiler is off, and 1 that it is on. The control input effectively renders the temperature evolution as a hybrid process that switches between two modes, one when the heater is on, and the other when it is off. So we can alternately think of the hybrid state space $\mathcal{S} = \mathbb{R} \times \{0, 1\}$.

Additionally, the temperature is measured and communicated to the controller by a sensor that is subject to noise, represented by $y_n = x_n + w_n$, with $\{w_n\}$ an i.i.d. sequence of Gaussian random variables with zero mean and variance

ω^2. The control input is assumed known, and hence we only consider the continuous observation process, $\mathcal{Y} = \mathcal{Y}^x = \mathbb{R}$.

A. Abstraction to POMDP

The discussed PBVI algorithm requires as input a POMDP with a finite set of states, observations, and control inputs. In order to apply Algorithm 2 to a partially observable hybrid system, we must first create a *finite state abstraction* of the original hybrid system, to approximately represent it as a POMDP as in Definition 1. We are interested in the use of abstractions with guaranteed behaviours, since the abstraction introduces an additional error into the evaluation of safety probabilities using the PBVI algorithm, because we are now evaluating safety probabilities over a *different* system. Formal abstractions are discussed in [26] and, for models with partial observations, in [4].

For safety critical systems, quantifying the probability that the system remains safe is crucial. When approximations are introduced, we must quantify the effect of the approximation on the safety probabilities being computed. We have already done this in Section IV-C, where the one step tolerance bound $\bar{\eta}$ is adjusted as a function of the error introduced by the PBVI algorithm, to ensure that the safety probability returned by Algorithm 2 is no more than a pre-specified distance η from the maximum. We can repeat this process to get a new one step tolerance $\bar{\eta}$ that also takes into account the error from the abstraction as in [4].

To create an abstraction of a partially observable stochastic hybrid system, the continuous spaces $\mathcal{X}$ and $\mathcal{Y}^x$ are partitioned into cells, and a representative point is assigned to each cell. For a certain class of stochastic hybrid systems, the error introduced by this partitioning, in combination with a PBVI algorithm, has been shown to be a linear function of the size of the partitions, and hence decreases to zero as the partitions are made smaller. A detailed description of the abstraction process, as well as the error it incurs, is provided in [4].

Here we will simply assume that the error introduced by the abstraction is known, and equal to δ^a, i.e. we know that $|V_0^B(\bar{R})$ - $V_{0,\delta}^B(\bar{R}_\delta)| \leq \delta^a$. The subscript δ indicates that the abstraction is being used (both when computing the value functions, and also because the probability densities for the original system must be mapped to probability mass functions, hence $\bar{R} \to \bar{R}_\delta$).

Proposition 3: For time horizon N, a POMDP abstraction $\mathcal{J}_\delta$ of a partially observable stochastic hybrid system $\mathcal{H}$ with abstraction error δ^a, and total desired tolerance η, if $\frac{\eta}{N} - \frac{2\delta^a}{N} - \delta^B \geq 0$, setting the one step tolerance to $\bar{\eta} = \frac{\eta - 2\delta^a}{N} - \delta^B$ ensures that the safety probability found by solving the lexicographic POMDP optimization using PBVI and the abstraction $\mathcal{J}_\delta$, denoted $V_{0,1,\delta}^{\eta,B}(\bar{R}_\delta)$ satisfies $p_{\text{safe}}^N(\pi^*, K; R) - V_{0,1,\delta}^{\eta,B}(\bar{R}) \leq \eta$.

VI. Applications to Smart Buildings

We now consider two case studies. The first is the finite state boiler maintenance problem introduced in II-B1, and the second is the hybrid state temperature regulation problem introduced in V-1. In each case we must utilize sensor measurements to design safe, cost-efficient control policies for building operation.

A. Optimised Boiler Maintenance

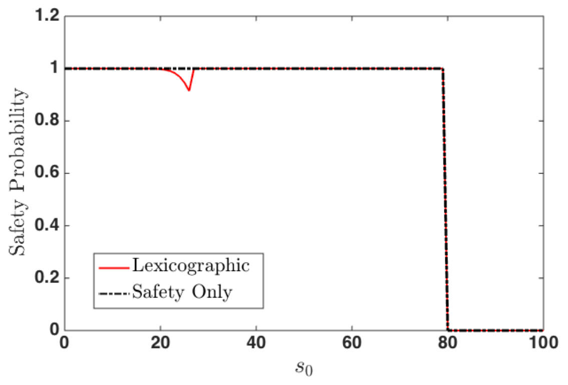
We apply the lexicographic approach in combination with PBVI to the boiler maintenance problem described in Sections II-B1 and III-1. We would like to maximise the probability that the boiler does not reach a degradation level of 80 or above, while simultaneously minimising the costs associated with maintaining and operating the boiler. To use Algorithm 2, we use inputs $K = \{0, \ldots, 79\}$, the cost function C given in (21), and the POMDP $\mathcal{J}$ presented in Section II-B1 extended to include binary variable q discussed in Section II-D. We consider a time horizon $N = 30$.

A set of 50 belief states B_n is generated sequentially for each time step. At initial time 0, the 50 belief states are initialised by randomly selecting the state in which the system starts with probability one, i.e. $b_0^i(s) = \mathbf{1}_{s^i}(s)$ with s^i selected uniformly at random from $\{0, \ldots, 79\}$ (we are not interested in starting from the unsafe set). The action u_0^i is also chosen at random, and an observation y^i is sampled from $\mathbb{P}(y|b_0^i, u_0^i)$ (9). The next belief state b_1^i is generated according to $M_{y,u}b$ in (8). This is done repeatedly for all $i = 1, \ldots, 50$ and $n = 1, \ldots, 30$.

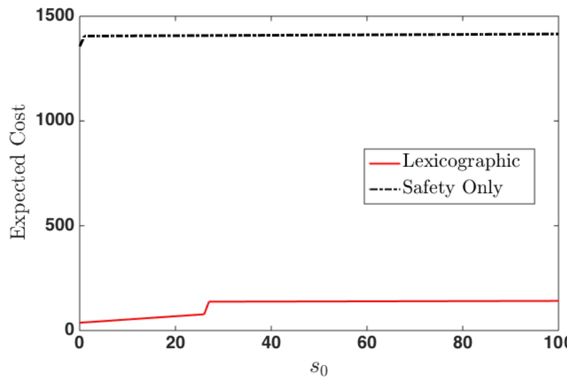
Next, the one step tolerance must be computed in order to guarantee that the overall tolerance is $\eta = 0.1$, i.e. that the probability that the boiler does not break down while minimising costs is not more than 0.1 below the maximum probability, obtained when costs are disregarded. To provide an exact bound on the tolerance requires incorporating the error δ^B from PBVI, which is difficult to compute and can be quite conservative. We therefore disregard the error δ^B, for two reasons. First, PBVI guarantees a lower bound to the actual value function when we use the lexicographic approach, and therefore although we do not have an exact guarantee on the distance from the optimal safety probability, we know that the safety probability is at least as high as the estimate we produce. Second, for this example PBVI returns safety probabilities that are equal to one, and therefore the PBVI solution is exact, since the probabilities are both lower and upper bounded by one, thus $\delta^B = 0$. Hence, we set $\bar{\eta} = \frac{\eta}{N} = \frac{1}{30}$.

The probability that the degradation level s_n does not reach 80 or above over the considered time horizon, given varying starting degradation levels s_0, is presented in Fig. 1a, using the lexicographic PBVI algorithm (Alg. 2) with a total tolerance $\eta = 0.1$. For comparison, the lexicographic PBVI algorithm is also solved using a tolerance $\eta = 0$, so that the safety objective is maximized. However, if the situation arises where either cleaning or not cleaning the boiler leads to the same safety probability (i.e. both are maximal) then the control input is selected to minimize costs (so u = NoClean is chosen over u = Clean). This highlights that the optimal control input is not necessarily unique – multiple control inputs may be optimal, and in that case we can set which to choose.

Similarly, Fig. 1b shows the expected costs over 30 days when safety is maximized without allowing for a trade-off to minimize costs ($\eta = 0$) versus allowing slack to also



(a)



(b)

Fig. 1: Comparison of safety probabilities (a) and expected total operational plus cleaning costs (b), over varying initial degradation levels s_0 and for $N = 30$ days, without considering the cost of fuel/cleaning (black, dashed line) and using Algorithm 2 to also minimize costs (red, solid line). The policy produced by maximizing safety alone leads to cleaning the boiler many times, whereas the lexicographic algorithm returns a policy that cleans the boiler at most once, with minimal effect on the probability of safety.

minimize costs ($\eta = 0.1$). Fig. 1b shows that the policy returned by Algorithm 2 with $\eta = 0.1$ is a threshold policy, since the expected cost increases by 100 (the cost of cleaning) for $s_0 > 26$, meaning that over 30 time steps, if the initial degradation level is below 26, we should not clean the boiler, and if it is greater than 26, the boiler should be cleaned only once. In a region below the threshold level of 26, there is a small sacrifice in the probability of safety (as seen in Fig. 1a) that does not exceed the tolerance $\eta = 0.1$, and is therefore deemed acceptable. If we were to shrink the tolerance level, the threshold degradation level at which cleaning should be performed would be lower ($s_0 < 26$). In contrast, when $\eta = 0$, although we select the control input that minimizes costs so long as the safety probability remains maximal, the expected cost is over 1000 regardless of the starting degradation level, hence the boiler is cleaned much more frequently.

We also test each policy (for $\eta = 0$ and $\eta = 0.1$) by simulating 1000 trajectories of the degradation level under each policy. Fig. 2 shows the control input (clean or do not clean) selected most frequently at each time step over the 1000 trajectories under each policy, starting from $s_0 = 30$. Under the safety maximizing policy, the boiler is cleaned at least every other day (which clearly does not make sense from a practical standpoint) whereas the lexicographic policy cleans the boiler once, on the first day, and does not clean it again.

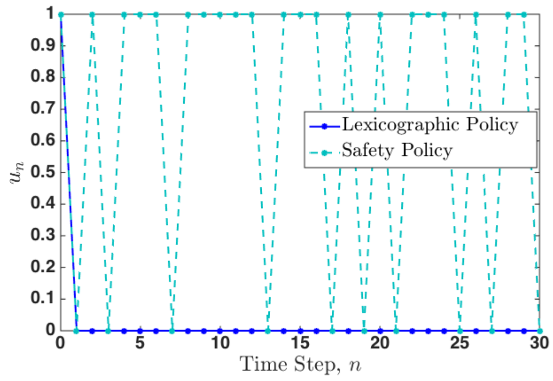


Fig. 2: Simulated policy averaged over 1000 trials, starting from $s_0 = 30$, generated using Algorithm 2 (blue, solid line) and PBVI with a single safety objective (turquoise, dashed line). When costs are not considered, the boiler is cleaned more than half of the time, versus a single cleaning using the lexicographic policy.

B. Parsimonious Temperature Regulation

Switching from boiler maintenance, consider now the parsimonious temperature regulation problem of Section V-1.

To use Algorithm 2, we first abstract the system to a finite state POMDP by discretizing $\mathcal{X}$ and $\mathcal{Y}^x$ (for details see [4]). We only need to discretize the set K rather than all of $\mathbb{R}$, because once the temperature leaves K, we are not interested in its actual value. For the observations, we discretize $\bar{K} = [16, 23.5]$, because the probability of observing y outside of $\bar{K}$ given $x \in K$ is $\epsilon \ll 1$ (specifically, $\epsilon = .0027$ for $\omega^2 = 0.25$). We use a grid spacing of $\delta^x = 0.01$ to create the finite set of states $\mathcal{S} = \{17.5, 17.51, \ldots, 21.99\}$, and a grid spacing of $\delta^y = 0.25$ to create the finite set of observations $\mathcal{Y} = \{16, 16.25, \ldots, 23.25\}$.

The probability distributions T and Y are constructed as for the boiler maintenance example, only now $T(s'|s,u)$ is derived from a Gaussian density function with mean $(1-a)s + bu + as_a$ and variance ν^2, and $Y(y|s)$ is derived from a Gaussian density with mean s and variance ω^2. For instance, $T(s'|s,u) = \int_{s'}^{s'+\delta^s} \phi(x; (1-a)s + bu + as_a, \nu^2)\, dx$ and similarly for Y (here $\phi(x; \mu, \sigma^2)$ is the Gaussian density with mean μ and variance σ^2 evaluated at x).

We compute the error bound δ^a on the abstraction using formulas derived in [25]. For $\delta^x = 0.01$, $\delta^y = .25$, and setting $\nu^2 = 0.2$, $\omega^2 = 0.25$, we obtain a total abstraction error over $N = 5$ time steps of $\delta^a = 0.135$. For a desired tolerance $\eta = 0.3$, the one step tolerance is $\bar{\eta} = 0.006$. We generate a set of 50 belief states in the same manner as the previous example, and apply Algorithm 2. Fig. 3a compares the maximal probability of safety (as underestimated by PBVI using tolerance $\eta = 0$) to the under-estimated safety probability when using

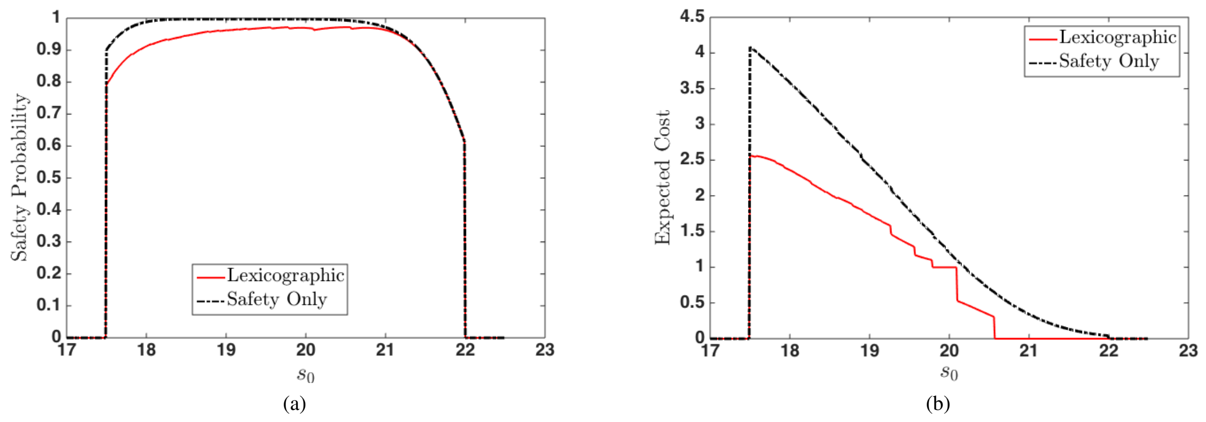


Fig. 3: Comparison of safety probabilities (a) and expected number of time steps the heater is turned on (b), over varying initial temperatures s_0 and for $N = 5$ time steps, without considering the cost of turning the heater on (black, solid line) and using Algorithm 2 to also minimize the number of time steps the heater is on (blue, dashed line). Allowing the probability of remaining within the desired temperature range to drop slightly can lead to using the heater almost half as often.

the lexicographic approach with $\eta = 0.3$, as a function of the initial temperature. Although the desired tolerance was made large in order to guarantee a non-zero one-step tolerance, Fig. 3a shows the actual distance from the maximal safety probability never exceeds 0.1 (because the error estimate δ^a is conservative). In fact, the safety probabilities produced by the lexicographic approach are quite close to the optimal, and even equal when the initial temperature is greater than 21, while the number of time steps the heater is on is lowered (cf. Fig. 3b). For an initial temperature between 17.5 and 18.5, the drop in likelihood of safety is around 0.1, while the number of time steps the heater is likely to be on drops by more than one. The approximate equality of the safety probabilities for $s_0 > 21$ is due to the short time horizon considered: in both cases there is no benefit to turning the heater on, because the probability of dropping below 17.5 is negligible over 5 time steps. The non-constant nature and the related adjacent discontinuities of the the expected cost around $s_0 = 20$ in Fig. 3b are likely because it is less straightforward to decide whether to turn the heater on or off, and because of the conflicting nature of the two objectives becomes more apparent.

C. Discussion on Computational Aspects

Constructing optimal policies for POMDPs is not easy [27], and even approximate algorithms such as point-based value iteration are time consuming. Further, the computation time increases with the number of states, observations, and control inputs that define the POMDP of interest. For a discussion on the complexity of PBVI, see [16]. A continuous or hybrid state system that is abstracted to a POMDP (as described in Section V) requires a large number of discretized states and observations in order for the abstraction error to be small, and as the dimension of the continuous state increases, the number of discretized states required can quickly become prohibitive.

The first case study we considered on optimal boiler maintenance had 101 states, 101 observations, and 2 control inputs. The computation time to produce the policy, safety probabilities, and expected cost presented in Fig. 1 for 30 time steps and 50 belief states was 128.34 seconds for the lexicographic approach with $\eta = 0.1$, run on a 2.7 GHz Intel Core i5 Macbook Pro with 8 GB of RAM. The temperature regulation case study, which had to be abstracted to a POMDP, had 550 states, 30 observations, and 2 control inputs. The computation time to produce the same results in Fig. 3 over 5 time steps and using 50 belief states was 541.77 seconds. The fivefold increase in the number of states had a far more significant impact on computation time than the reduction in the number of time steps between the two case studies.

In comparison, however, the overall computation time to maximize safety without allowing any slack to minimize costs was significantly greater in both cases. To generate the optimal policy, maximal safety probability, and expected cost for the boiler maintenance problem with $\eta = 0$ took 1646.25 seconds, and for the temperature regulation problem took 1344.43 seconds.

In both cases, the number of α-vectors stored in Γ_n for all n was greater than with a tolerance $\eta > 0$, indicating that the number of redundant α-vectors that corresponded to the same optimal policy tree for different belief states was fewer for $\eta = 0$. Fewer α-vectors leads to less computation at each time step. As such, the computation time was longer for the boiler maintenance problem, due to the additional number of α-vectors stored and computed over a higher number of time steps. The increased number of states in the temperature regulation problem no longer outweighed the increase in the time horizon of interest with the greater number of α-vectors.

Hence when slack is introduced to simultaneously minimize costs, the total number of control inputs to choose from is restricted, and the likelihood of having redundant α-vectors at each time step increases. While we cannot claim that the lexicographic approach will lead to faster computation time in all cases, we did find in both of the examples presented that the computation time increased as η decreased. It seems reasonable to expect that in general, not only does the lexicographic algorithm allow a precise trade-off between opposing optimization criteria, but it also can decrease computation time required for the PBVI algorithm.

VII. Conclusions and Future Work

With focus on partially observable stochastic models, possibly with continuous and hybrid dynamics, this work has discussed the application of a lexicographic framework for multi-objective optimisation to problems with safety requirements, thus integrating correct-by-design synthesis for safety and optimal synthesis for performance. Computationally, the work has leveraged the use of compact representation of belief states, of sampling-based techniques (PBVI), and of formal abstractions, and showcased the results on two case studies in the area of smart buildings.

Methodologically, we are interested in exploring further connections with algorithms for approximate dynamic programming, and in furthering the usability of error bounds for formal abstractions [25], [26]. In the area of modeling and control for smart buildings, we are interested in extensions to hybrid models of boiler degradation, and in developing and employing realistic models derived from real data.

References

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[2] J. Ding and C. Tomlin, "Robust reach-avoid controller synthesis for switched nonlinear systems," in *IEEE Conference on Decision and Control*, 2010, pp. 6481–6486.

[3] K. Lesser and M. Oishi, "Reachability for partially observable discrete time stochastic hybrid systems," *Automatica*, vol. 50, no. 8, pp. 1989–1998, 2014.

[4] ——, "Approximate ssfety verification and control of partially observable stochastic hybrid systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 81 – 96, 2017.

[5] K. Dräger, V. Forejt, M. Z. Kwiatkowska, D. Parker, and M. Ujma, "Permissive controller synthesis for probabilistic systems," in *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014*, 2014, pp. 531–546.

[6] W. Kuijper and J. van de Pol, "Compositional control synthesis for partially observable systems," in *CONCUR 2009 - Concurrency Theory*, ser. Lecture Notes in Computer Science, M. Bravetti and G. Zavattaro, Eds. Springer Berlin Heidelberg, 2009, vol. 5710, pp. 431–447.

[7] X. Yin and S. Lafortune, "Synthesis of maximally permissive nonblocking supervisors for partially observed discrete event systems," in *IEEE Conference on Decision and Control*, 2014, pp. 5156–5162.

[8] M. Svorenova, I. Cerna, and C. Belta, "Optimal control of MDPs with temporal logic constraints," in *Proceedings of the 52nd IEEE Conference on Decision and Control, CDC 2013, December 10-13, 2013, Firenze, Italy*, 2013, pp. 3938–3943.

[9] T. Chen, V. Forejt, M. Z. Kwiatkowska, A. Simaitis, and C. Wiltsche, "On stochastic games with multiple objectives," in *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013*, 2013, pp. 266–277.

[10] L. Zhang, H. Hermanns, and D. N. Jansen, "Logic and model checking for hidden Markov models," in *Proc. on Formal Techniques for Networked and Distributed Systems*. Springer, 2005, pp. 98–112.

[11] S. Giro and M. N. Rabe, "Verification of partial-information probabilistic systems using counterexample-guided refinements," in *Proc. on Automated Technology for Verification and Analysis*, ser. LNCS. Springer, 2012, pp. 333–348.

[12] K. Chatterjee, M. Chmelik, and M. Tracol, "What is decidable about partially observable Markov decision processes with omega-regular objectives," in *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, 2013, pp. 165–180.

[13] L. G. Mitten, "Preference order dynamic programming," *Management Science*, vol. 21, no. 1, pp. 43–46, 1974.

[14] K. H. Wray, S. Zilberstein, and A.-I. Mouaddib, "Multi-objective MDPs with conditional lexicographic reward preferences," in *International Conference on Artificial Intelligence*, 2015, pp. 3418–3424.

[15] K. H. Wray and S. Zilberstein, "Multi-objective POMDPs with lexicographic reward preferences," in *International Joint Conference on Artificial Intelligence*, 2015, to Appear.

[16] J. Pineau, G. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *Journal of Artificial Intelligence Research*, vol. 27, pp. 335–380, 2006.

[17] R. Mobley, *An Introduction to Predictive Maintenance*, ser. Plant Engineering. Elsevier Science, 2002.

[18] (2012) Studies show: Hvac system maintenance saves energy. Institute for Building Efficiency, An Initiative of Johnson Controls. 444 North Capitol St., NW Suite 729, Washington DC 20001.

[19] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2005, vol. 1.

[20] J. Dupačová and K. Sladký, "Comparison of multistage stochastic programs with recourse and stochastic dynamic programs with discrete time," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 82, no. 11-12, pp. 753–765, 2002.

[21] J. Berka and K. Macek, "Effective maintenance of stochastic systems via dynamic programming," in *Proceedings of 19th Technical Computing Prague Conference*, Prague, Czech Republic, 2011.

[22] E. Sondik, "The optimal control of partially observable Markov processes," Ph.D. dissertation, Stanford University, 1971.

[23] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[24] J. Ding, A. Abate, and C. Tomlin, "Optimal control of partially observable discrete time stochastic hybrid systems for safety specifications," in *American Control Conference*, 2013, pp. 6231–6236.

[25] K. Lesser and M. Oishi, "Finite state approximation for verification of partially observable stochastic hybrid systems," in *Hybrid Systems: Computation and Control*, 2015.

[26] S. Soudjani and A. Abate, "Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes," *SIAM Journal on Applied Dynamical Systems*, vol. 12, no. 2, pp. 921–956, 2013.

[27] C. Lusena, J. Goldsmith, and M. Mundhenk, "Nonapproximability results for partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 14, pp. 83–103, 2001.

Appendix

Proof of Proposition 1: We first show by induction that $V_n^*(\bar{b}) - V_{n,1}^{\eta}(\bar{b}) \leq (N-n)\bar{\eta}$ for any $\bar{b} \in \mathfrak{B}$, with V_n^* the value function that returns the true maximal safety probability, i.e. $V_0^*(\bar{R}) = p_{\text{safe}}^N(\pi^*, K; R)$. At time N, $V_N^*(\bar{b}) = \sum_{\bar{s}} \mathbf{1}_{K\times\{1\}}(\bar{s})\bar{b}(\bar{s})$, which does not depend on the control input, and so $V_N^*(\bar{b}) = V_{N,1}^{\eta}(\bar{b})$ for all $\bar{b}$, and the result is satisfied. Next assume $V_{n+1}^*(\bar{b}) - V_{n+1,1}^{\eta}(\bar{b}) \leq (N-n-1)\bar{\eta}$. At time n,

$$V_n^*(\bar{b}) - V_{n,1}^{\eta}(\bar{b}) = V_n^*(\bar{b}) - V_{n,1}^{u^*}(\bar{b}) + V_{n,1}^{u^*}(\bar{b}) - V_{n,1}^{\eta}(\bar{b})$$

with $V_{n,1}^{u^*}$ the value function that uses the same optimal control input u^* at time n as V_n^*, but is defined recursively through the lexicographic value function $V_{n,1}^{\eta}$ (this is the same notation as used in the definition of $\mathcal{U}_{N,i}(\bar{b})$ (22)). Then,

$$\begin{aligned}
V_n^*(\bar{b}) - V_{n+1}^{\eta}(\bar{b}) &\leq V_n^*(\bar{b}) - V_{n,1}^{u^*}(\bar{b}) + \bar{\eta} \\
&\leq \sum_y V_{n+1}^*(\bar{M}_{y,u^*}\bar{b})\mathbb{P}(y|\bar{b},u^*) \\
&\qquad - \sum_y V_{n+1,1}^{\eta}(\bar{M}_{y,u^*}\bar{b})\mathbb{P}(y|\bar{b},u^*) + \bar{\eta} \\
&\leq (N-n-1)\bar{\eta}\sum_y \mathbb{P}(y|\bar{b},u^*) + \bar{\eta} \\
&\leq (N-n)\bar{\eta}.
\end{aligned}$$

The first line follows from the definition of the one step tolerance (see (22)), the second to last line by the induction hypothesis, and the last line because $\sum_y \mathbb{P}(y|\bar{b},u) = 1$. Since $V_0^*(\bar{b}) - V_{0,1}^{\eta}(\bar{b}) \leq N\bar{\eta}$ for all $\bar{b}$, we can achieve $V_0^*(\bar{b}) - V_{0,1}^{\eta}(\bar{b}) \leq \eta$ by setting $\bar{\eta} \leq \frac{\eta}{N}$. ■

Proof of Proposition 2: First, because the PBVI algorithm is designed to produce a value function that is a lower bound on the actual value function, $V_{0,1}^{\eta,B} \leq V_{0,1}^{\eta}$. Further, by (32), we know that $V_0^*(\bar{b}) \leq N\delta^B + V_0^B(\bar{b})$. Then,

$$\begin{aligned}
V_0^*(\bar{R}) - V_{0,1}^{\eta}(\bar{R}) &\leq V_0^*(\bar{R}) - V_{0,1}^{\eta,B} \\
&\leq V_0^B(\bar{R}) + N\delta^B - V_{0,1}^{\eta,B}(\bar{R}).
\end{aligned}$$

Next, because we are enforcing the one step tolerance $\bar{\eta}$ between $\max_{u\in\mathcal{U}} V_{n,1}^{\eta,B}(\bar{b})$ and $V_{n,1}^{\eta,B}(\bar{b})$, we can use the same argument as in the proof of Prop. 1 to show that $V_0^B(\bar{R}) - V_{0,1}^{\eta,B}(\bar{R}) \leq N\bar{\eta}$. Therefore

$$V_0^*(\bar{R}) - V_{0,1}^{\eta,B}(\bar{R}) \leq N\delta^B + N\bar{\eta}$$

and we can guarantee that $V_0^*(\bar{R}) - V_{0,1}^{\eta,B}(\bar{R}) \leq \eta$ by setting $\bar{\eta} \leq \frac{\eta}{N} - \delta^B$. Intuitively, as the error from PBVI decreases to zero, the one step tolerance approaches that required in the case of having an exact solution. ■

Proof of Proposition 3: As in the proof of Proposition 2, recall that $V_{0,1}^{\eta,B} \leq V_{0,1}^{\eta}$ and $V_0^*(\bar{b}) \leq N\delta^B + V_0^B(\bar{b})$. Also note that when we enforce the one step tolerance, we are enforcing it between $\max_{u\in\mathcal{U}} V_{n,1,\delta}^{\eta,B}$ and $V_{n,1,\delta}^{\eta,B}$, and therefore by the same argument as for Proposition 1, $V_{0,\delta}^B(\bar{R}) - V_{0,1,\delta}^{\eta,B}(\bar{R}) \leq N\bar{\eta}$. Then,

$$\begin{aligned}
V_0^*(\bar{R}) - V_{0,1}^{\eta}(\bar{R}) &\leq V_0^B(\bar{R}) + N\delta^B - V_{0,1}^{\eta,B}(\bar{R}) \\
&\leq V_0^B(\bar{R}) + N\delta^B - V_{0,\delta}^B(\bar{R}) + V_{0,\delta}^B(\bar{R}) \\
&\quad - V_{0,1,\delta}^{\eta,B}(\bar{R}) + V_{0,1,\delta}^{\eta,B}(\bar{R}) - V_{0,1}^{\eta,B}(\bar{R}) \\
&\leq \left|V_0^B(\bar{R}) - V_{0,\delta}^B(\bar{R})\right| + V_{0,\delta}^B(\bar{R}) - V_{0,1,\delta}^{\eta,B}(\bar{R}) \\
&\quad + \left|V_{0,1,\delta}^{\eta,B}(\bar{R}) - V_{0,1}^{\eta,B}(\bar{R})\right| + N\delta^B \\
&\leq \delta^a + N\bar{\eta} + \delta^a + N\delta^B
\end{aligned}$$

Hence $V_0^*(\bar{R}) - V_{0,1}^{\eta}(\bar{R}) \leq \eta$ if we can enforce $\bar{\eta} \leq \frac{\eta - 2\delta^a}{N} - \delta^B$, which is true if $\frac{\eta-2\delta^a}{N} - \delta^B \geq 0$. ■

Kendra Lesser received her Ph.D. in Electrical Engineering in 2014, and an M.S. in Mathematics in 2010 from the University of New Mexico. She received her B.Sc. in Mathematics from McGill University in 2007. She is currently a Senior Controls Engineer at Verus Research in Albuquerque, NM, and prior to that was a Marie Curie Research Fellow in the Computer Science Department at the University of Oxford. Her main research interests include stochastic optimization, planning under uncertainty, verification, and learning algorithms, with a focus on safe, provably correct planning and estimation. Applications include optimised maintenance and control of building heating systems, motion planning algorithms, and guidance and control of spacecraft.

Alessandro Abate (S'02–M'08) is an Associate Professor in the Department of Computer Science at the University of Oxford (UK), and is additionally affiliated with the Alan Turing Institute in London (UK). He received a Laurea in Electrical Engineering in October 2002 from the University of Padova (IT), an MS in May 2004 and a PhD in December 2007, both in Electrical Engineering and Computer Sciences, at UC Berkeley (USA). He has been an International Fellow in the CS Lab at SRI International in Menlo Park (USA), and a PostDoctoral Researcher at Stanford University (USA), in the Department of Aeronautics and Astronautics. From June 2009 to mid 2013 he has been an Assistant Professor at the Delft Centre for Systems and Control, TU Delft - Delft University of Technology (NL).
His research interests are in the analysis, verification, and control of probabilistic and hybrid models, and in their general application over a number of domains, particularly in energy and in systems biology.