Robust Parameter Learning for Uncertain MDPs

Yannik Schnitzer, Alessandro Abate, David Parker

University of Oxford, Oxford, United Kingdom

Abstract. Learning-based approaches to verifying unknown Markov decision processes (MDPs) often employ uncertain MDPs. These models use, for example, confidence intervals to capture transition uncertainty and allow synthesis of policies that are robust to this uncertainty. However, this approach typically quantifies uncertainty independently for individual transition probabilities, ignoring dependencies due to shared latent quantities. We propose to learn such models using parametric MDPs (pMDPs), where transition probabilities are expressions over a set of parameters. We project statistical uncertainty from empirical transition frequencies onto the pMDP's parameter space, yielding a probably approximately correct (PAC) uncertainty model for the underlying MDP that respects the algebraic dependencies between transitions. The resulting models are algorithmically challenging to solve, so we propose a hierarchy of sound polytopic outer approximations of the induced confidence set. We implement and evaluate our approach, demonstrating substantially tighter uncertainty estimates than classical interval-based uncertain MDP learning techniques.

1 Introduction

Markov decision processes (MDPs) are the standard formalism for sequential decision-making under uncertainty. Their classical formulation assumes that all transition probabilities are known exactly, which is rarely realistic in practice. Two closely related model classes relax this assumption: *uncertain MDPs* (UMDPs) [23, 33, 47] and *parametric MDPs* (pMDPs) [21, 28]. UMDPs represent *epistemic* uncertainty by associating transition probabilities with *uncertainty sets* of admissible values; interval MDPs (IMDPs) are a common example. pMDPs specify transition probabilities symbolically as expressions over parameters that range over an admissible parameter space. Both formalisms thus capture imprecise model knowledge by describing not a single precise model, but a family of models, each representing a possible true underlying system.

Parametric MDPs have found applications in model repair [7, 30], sensitivity analysis [5], and quality-of-service control [12], predominantly via the problem of *parameter synthesis* [13, 16, 17, 24, 36], which partitions the parameter space into two regions, corresponding to parameter valuations for which the model respectively satisfies or violates a given property. Uncertain MDPs have emerged as a standard model for reasoning about *robustness*: they support the synthesis of *robust* policies, i.e., policies that are optimal under the assumption that epistemic uncertainty is resolved adversarially by the environment, selecting the worst-case admissible distributions with respect to the agent's desired objective [23, 33, 47].

UMDPs are therefore well suited to *learning-based* methods for the verification of MDPs whose transition probabilities are unknown. Typically, uncertainty sets for transition probabilities are learned from interactions with the true but unknown system, for instance as statistically sound confidence sets such as confidence intervals [2, 32, 38, 40, 43]. The resulting learned UMDP then contains the true system with high confidence. Consequently, any robust policy synthesised on this UMDP is guaranteed, with the same confidence, to achieve on the true system at least the robust value certified by the learned model.

Existing approaches to UMDP learning typically treat transition probabilities across the model as independent, learning separate confidence sets for each of them [6, 42, 43]. In practice, however, these distributions are often coupled through shared latent quantities, such as common failure probabilities, shared reliability parameters or global environmental conditions that remain fixed and affect multiple transitions. Such dependencies are often structurally known and are conveniently represented by pMDPs, even though the precise values of the underlying quantities are unknown and cannot be directly observed.

This work addresses the problem of UMDP learning under known structural parametric dependencies that are captured by a pMDP. We assume a known pMDP for which an unknown parameter instantiation induces the true underlying system. From sampled interactions with this system, we first derive statistical confidence sets for the individual transition probabilities and then exploit the parametric structure of the pMDP to project these sets into the parameter space. This yields a confidence region in parameter space that contains the true underlying parameter instantiation with high confidence.

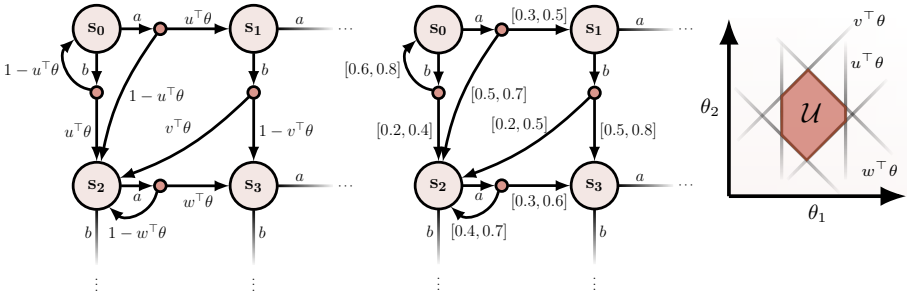
Example. We illustrate this idea using an example: a Mars-rover navigation domain, inspired by the semi-autonomous vehicle environment in [25, 41]. A controller communicates with the rover via two lossy satellite channels while the rover attempts to collect samples and return home. The probability of successful communication depends on the rover's position through known coefficients, but on unknown channel reliabilities. At each position, it is given by $v^\top \theta = v_1\theta_1 + v_2\theta_2$, where $v = (v_1, v_2)$ is position-dependent and $\theta = (\theta_1, \theta_2)$ are the unknown reliabilities. The domain is therefore naturally modelled as a pMDP as shown in Figure 1a.

From sampled transitions of the rover, the standard approach learns an IMDP by assigning a confidence interval to each transition probability; see Figure 1b. This yields a robust model with formal confidence guarantees on inclusion of the true unknown system, but treats transition probabilities as independent. In our example, however, they are coupled through the shared parameter instantiation θ.

Our approach exploits this structure by projecting the learned transition-wise intervals through the parametric structure of the pMDP into the parameter space. Each learned transition confidence interval $[l, u]$ induces constraints of the form

$$l \leq v_1\theta_1 + v_2\theta_2 \leq u.$$

Collecting these constraints yields a parameter confidence region $\mathcal{U}$, as shown in Figure 1c. With high confidence, the true parameter instantiation lies in $\mathcal{U}$.



(a) Parametric MDP (b) Learned IMDP (c) Confidence Region

Fig. 1: Construction of the parameter confidence region $\mathcal{U}$ from a learned IMDP.

The confidence region $\mathcal{U}$, together with the pMDP, induces a UMDP U that contains the true MDP with high confidence. This is tighter than a UMDP constructed from the confidence sets for individual transitions (e.g., the IMDP in Fig. 1b) since it rules out transition functions that are inconsistent with any of the constraints induced by the parametric expressions and the learned intervals, leading to less conservative robust policies and tighter guarantees from the same data.

However, because of its (non-rectangular) structure, the UMDP U is computationally very challenging to solve. We tackle this problem by proposing a hierarchy of relaxations of the induced uncertainty set, yielding UMDPs that can be solved more efficiently to generate robust policies. All relaxations are sound and thus produce *probably approximately correct* (PAC) [44] performance guarantees. This hierarchy illustrates a clear trade-off between the tightness and efficiency of the relaxations. We implement our methods and evaluate them on a range of established benchmarks. The results demonstrate substantially tighter uncertainty estimates than classical interval-based uncertain MDP learning techniques.

2 Preliminaries

For a finite set X, let $\Delta(X) = \{\mu \colon X \to [0,1] \mid \sum_{x \in X} \mu(x) = 1\}$ denote the set of probability distributions over X.

Definition 1 (Markov decision process). *A (finite) MDP is a tuple $M = (S, A, s_0, P)$, where S is a finite set of states, A is a finite set of actions, $s_0 \in S$ is an initial state, and $P : S \times A \to \Delta(S)$ is a transition kernel. We write $P(s,a)(s')$ (or $P(s,a,s')$) for the probability to move from state s to state s' under action a.*

A *policy* (or *strategy*) is a function $\pi \colon S \to A$ that selects an action $\pi(s)$ in each state s. An (infinite) *path* is a sequence $\rho = s^0 a^0 s^1 a^1 \cdots \in (S \times A)^\omega$ such that $P(s^t, a^t)(s^{t+1}) > 0$ for all $t \geq 0$. We write $\mathrm{Paths}(s)$ for the set of paths starting in s and Π for the set of policies. An *objective* is a measurable mapping $R \colon \mathrm{Paths}(s) \to \mathbb{R}$ assigning a return to each path. Typical objectives are based on satisfaction of temporal-logic properties [10] or accumulation of rewards [35].

A policy $\pi \in \Pi$ executed in an MDP M induces a probability measure over Paths(s) [35]. We define the *value* of π in s as $V_M^\pi(s) \coloneqq \mathbb{E}_{M,s}^\pi[R]$.

2.1 Parametric and Uncertain MDPs

Let $\Theta = \{\theta_1, \ldots, \theta_\ell\}$ be an ordered set of parameters. A *parameter instantiation* is a vector $\mathbf{u} \in \mathbb{R}^\ell$. We assume that all parameters are bounded, i.e., $l_i \leq \mathbf{u}_i \leq u_i$ for some $l_i, u_i \in \mathbb{R}$, and refer to $\mathcal{D} \coloneqq \prod_{i=1}^{\ell}[l_i, u_i]$ as the *parameter space*. Let $\mathbb{Q}[\Theta]$ denote the ring of polynomials over Θ with rational coefficients. Each $f \in \mathbb{Q}[\Theta]$ induces a function $f : \mathbb{R}^\ell \to \mathbb{R}$, where $f[\mathbf{u}]$ denotes the polynomial f evaluated by substituting every occurrence of θ_i with its instantiation $\mathbf{u}_i$. We denote by

$$\Delta_\Theta(X) = \{f \colon X \to \mathbb{Q}[\Theta] \mid \forall \mathbf{u} \in \mathcal{D}.\ f(x)[\mathbf{u}] \in [0,1], \sum_{x \in X} f(x)[\mathbf{u}] = 1\}$$

the set of all *parametric probability distributions* over a finite set X, i.e., the set of all parametric functions that induce a valid probability distribution over X for any parameter instantiation $\mathbf{u} \in \mathcal{D}$.

Parametric Markov decision processes extend standard MDPs by allowing transition probabilities to be specified as parametric probability distributions, where fixing a parameter instantiation yields an ordinary MDP.

Definition 2 (Parametric MDP). *A* parametric MDP *(pMDP) is a tuple $M_\Theta = (S, A, s_0, \Theta, P_\Theta)$, where S, A, s_0 are as for MDPs, Θ is a parameter set, and $P_\Theta : S \times A \to \Delta_\Theta(S)$ is a* parametric *transition probability function. For a parameter instantiation $\mathbf{u} \in \mathbb{R}^l$, we obtain the instantiated MDP $M[\mathbf{u}] = (S, A, s_0, P[\mathbf{u}])$ with $P[\mathbf{u}](s,a)(s') \coloneqq P_\Theta(s,a)(s')[\mathbf{u}]$, for all $s, s' \in S,\ a \in A$.*

In this work, we assume there is a true but unknown MDP M, obtained by instantiating a *known* pMDP M_Θ with an *unknown* parameter instantiation $\mathbf{u}$. The objective is to learn, from sampled interactions, an approximation of the true dynamics with quantified uncertainty, and to synthesise policies that are robust to the residual uncertainty. To represent partial knowledge, we adopt the framework of UMDPs. A UMDP is an MDP in which transition probabilities are not fixed, but are only required to lie in specified *uncertainty sets*.

Definition 3 (Uncertain MDP). *A UMDP is a tuple $U = (S, A, s_0, \mathcal{P})$, where S, A, s_0 are as for MDPs and $\mathcal{P} \subseteq (\Delta(S))^{S \times A}$ is an* uncertainty set *of admissible transition probability functions $P \in \mathcal{P}$. The UMDP is* (s,a)-rectangular *if $\mathcal{P}$ factorises as the product of local uncertainty sets, i.e., $\mathcal{P} = \prod_{(s,a) \in S \times A} \mathcal{P}(s,a)$, where $\mathcal{P}(s,a) \subseteq \Delta(S)$. Otherwise, the uncertainty is* coupled *(non-rectangular).*

There is a close relationship between pMDPs and UMDPs. Given a pMDP and a set $\mathcal{U}$ of admissible parameter instantiations, we obtain a UMDP by letting the uncertainty set contain exactly the successor distributions induced by instantiations $\mathbf{u} \in \mathcal{U}$. Conversely, a UMDP can be represented as a pMDP together with a set of admissible parameter instantiations inducing precisely the uncertainty set. We formalise this correspondence in Appendix B.

In this work, the known pMDP serves as a structural model, capturing dependencies across transitions. From sampled interactions with the unknown system, we infer a set of plausible parameter instantiations $\mathcal{U}$ with quantified confidence and, equivalently, a UMDP whose uncertainty sets contain the transition distributions consistent with $\mathcal{U}$. This UMDP is then used for robust policy synthesis under the remaining uncertainty about the true dynamics of M.

Interval MDPs. We recall a standard class of UMDPs that we use throughout. An *interval MDP* (IMDP) specifies, for each $(s, a, s') \in S \times A \times S$, componentwise lower and upper bounds $\ell_{(s,a,s')}, u_{(s,a,s')} \in [0, 1]$. The admissible successor distributions from (s, a) then are those whose components respect these bounds:

$$\mathcal{P}(s,a) \;=\; \Big\{ \mu \in \Delta(S) \;\Big|\; \forall s' \in S :\; \ell_{(s,a,s')} \le \mu(s') \le u_{(s,a,s')} \Big\}.$$

2.2 Solving Uncertain MDPs

For a UMDP, a *robust* policy is one that optimises worst-case performance over all admissible transition kernels. Following the standard robust MDP interpretation [23, 33, 47], we view uncertainty as being resolved adversarially by an *environment* (or *nature*) that selects admissible transition probabilities. Formally, an agent policy π and an admissible transition kernel $P \in \mathcal{P}$ induce a probability measure over paths [48]. We denote the corresponding expectation by $\mathbb{E}^{\pi}_{P,s}[R]$ and define the value of state s under (π, P) as $V^{\pi}_{P}(s) \coloneqq \mathbb{E}^{\pi}_{P,s}[R]$. The *robust value* is the maximal expected return under the worst-case nature:

$$V^{*}_{U}(s) \;\coloneqq\; \sup_{\pi \in \Pi} \; \inf_{P \in \mathcal{P}} \; V^{\pi}_{P}(s), \tag{1}$$

and a *robust-optimal policy* is any $\pi^* \in \operatorname{argsup}_{\pi} \inf_{P} V^{\pi}_{P}(s)$. In particular, π^* guarantees to achieve at least $V^{*}_{U}(s)$ for any admissible transition kernel $P \in \mathcal{P}$.

Computing (1) can be challenging in general, in particular for coupled uncertainty sets [47]. For *rectangular* uncertainty (i.e. $\mathcal{P} = \prod_{(s,a)} \mathcal{P}(s, a)$), the inner minimisation decomposes into local optimisations at each state–action pair, and the robust value satisfies a Bellman-type optimality equation. For example, for reachability with target set $T \subseteq S$, the robust Bellman equations take the form

$$V^{*}_{U}(s) = \max_{a \in A} \underbrace{\min_{\mu \in \mathcal{P}(s,a)} \sum_{s' \in S} \mu(s')\, V^{*}_{U}(s')}_{\text{Inner Optimisation}}, \quad s \in S \setminus T, \tag{2}$$

together with boundary conditions $V^{*}_{U}(s) = 1$ for all $s \in T$. Analogous robust Bellman equations can be given, e.g., for expected or discounted reward objectives [23, 33]. These equations can be solved with *robust value iteration*, i.e., by repeatedly applying the robust Bellman operator until convergence [23, 33].

2.3 Learning Uncertain MDPs

We review the basics of learning an UMDP from data generated by an unknown MDP M. We assume access to a dataset of transition samples $C = \{(s_i, a_i, s'_i)\}_{i=1}^{n}$,

obtained either (i) by collecting trajectories in the MDP, i.e., episodic sampling with restarts from the initial state, or continuing interaction, or (ii) via a *generative model* that allows direct sampling of next states for state–action pairs [40, 42].

From C we extract the empirical counts $\#(s,a)$ and $\#(s,a,s')$ for all $s, s' \in S,\ a \in A$, where $\#(s,a)$ is the number of occurrences of (s,a) and $\#(s,a,s')$ the number of observed transitions from s to s' under action a. Whenever $\#(s,a) > 0$, the empirical estimate of the transition probability is $\tilde{P}(s,a,s') := \frac{\#(s,a,s')}{\#(s,a)}$.

A standard UMDP learning technique is to construct an IMDP that over-approximates the unknown transition probabilities by confidence intervals. For each (s,a,s'), we view $\#(s,a,s')$ as a binomial observation with $\#(s,a)$ trials and success probability $P(s,a,s')$, and lift the empirical estimate $\tilde{P}(s,a,s')$ to a confidence interval $I_\gamma(s,a,s') := [\,\ell(s,a,s'),\, u(s,a,s')\,]$, such that $\Pr\big[P(s,a,s') \in I_\gamma(s,a,s')\big] \geq 1-\gamma$, for $\gamma \in (0,1)$. Suitable choices for I_γ include standard binomial confidence intervals such as the Clopper–Pearson interval [14, 32]. If $\#(s,a) = 0$, no information is available for that state–action pair, and we therefore use the trivial interval $I_\gamma(s,a,s') = [0,1]$ for all $s' \in S$.

For the objectives considered in this paper, such as reachability probabilities, intervals containing both zero and non-zero values can be handled directly by the robust value iteration in Eq. (2). For more general temporal-logic objectives, however, satisfaction may depend on the graph structure of the UMDP. In that case, intervals containing both zero and non-zero values allow the adversarial nature to alter the graph structure itself, which complicates robust synthesis. This issue is not specific to our approach, but inherent to temporal-logic reasoning over UMDPs in general. Existing techniques therefore often assume *constant support*, i.e., that the set of successors is known a priori [42, 43, 48]. Under this assumption, one may replace any lower bound 0 in the learned intervals by a sufficiently small $\varepsilon > 0$ in order to preserve the known graph structure.

To obtain an overall confidence level $1-\delta$, with $\delta \in (0,1)$, we distribute the confidence budget across all transition probabilities and set $\gamma := \delta / |S \times A \times S|$. Hence, with probability at least $1-\delta$, the unknown MDP is contained in the learned IMDP (e.g., [38, 43]). Consequently, the value of a policy on the learned IMDP yields a $(1-\delta)$-valid lower bound on the performance of the policy in M. Other common constructions of uncertainty sets $\mathcal{P}(s,a)$ with formal confidence guarantees include, e.g., L_1-norm balls based on the Weissman inequality [40, 46].

2.4 Parameter Tying

An established method for improving IMDP learning in the presence of known parametric dependencies is *parameter tying* [18, 34, 37]. Consider two transitions (s,a,s') and (t,b,t') in M_Θ with $s \neq t$ and $P_\Theta(s,a,s') = P_\Theta(t,b,t')$, i.e., both transition probabilities are described by the same expression. Then the samples in C obtained for these transitions correspond to the same Bernoulli experiment and can be *pooled* to derive a single interval, which is then assigned to both transitions. Moreover, the number of components to be learned over which we distribute the overall confidence budget δ reduces to the number of distinct parametric expressions in M_Θ. Details are provided in Appendix C.

Parameter tying applies only when parametric expressions are *exactly* equivalent. Our approach instead aggregates information from the parametric structure and the collected samples by projecting the learned constraints through the algebraic structure of M_Θ into the parameter space. This jointly accounts for observations across all transitions and can tighten uncertainty even for transitions whose expressions have not been observed in any sample in C. We use parameter tying as a pre-processing step: we pool the observed counts for equivalent parametric expressions and then apply our technique to the pooled data. Let

$$\Lambda \;=\; \{\, P_\Theta(s,a,s') \mid s,s' \in S,\; a \in A \,\} \tag{3}$$

be the set of distinct parametric expressions occurring in M_Θ. We distribute the confidence budget only across those expressions that are not constant polynomials: $\Lambda_{\mathsf{unk}} \;:=\; \{\, f \in \Lambda \mid f \notin \mathbb{Q} \,\}$. For each $f \in \Lambda_{\mathsf{unk}}$, we compute from the pooled counts a confidence interval $[l_f, u_f]$ as per Section 2.3 such that, under the true but unknown parameter instantiation $\mathbf{u}$, $\Pr\big[\, l_f \le f[\mathbf{u}] \le u_f \,\big] \;\ge\; 1 - \frac{\delta}{|\Lambda_{\mathsf{unk}}|}$. For constant expressions $f \in \Lambda \setminus \Lambda_{\mathsf{unk}}$, we set $l_f = u_f = f$. Substituting each occurrence of f in M_Θ by the corresponding interval $[l_f, u_f]$ yields an IMDP that contains the true (unknown) MDP M with probability at least $1-\delta$.

3 Robust Parametric UMDP Learning

IMDP learning inherently assumes that all transition probabilities are *independent* and learned as individual confidence intervals. We now introduce our approach which leverages dependencies specified by an underlying pMDP model. From the observed transitions, we infer a set of plausible instantiations $\mathcal{U}$ with quantified confidence, exploiting the dependencies captured by shared parameters.

Throughout, we consider a fixed pMDP M_Θ and an unknown MDP M induced by M_Θ under an unknown parameter instantiation $\mathbf{u}$. We further fix a data set of i.i.d. transition samples C obtained from M, as described in Section 2.3. We focus initially on the case where the per-transition uncertainty sets are intervals, i.e., we learn an IMDP approximation of M and then project it into the parameter space. The same ideas extend to other classes of uncertainty sets, such as L_1-balls that preserve per-state coupling [40], as we shall discuss in Section 3.4.

Given the learned intervals $[l_f, u_f]$ for $f \in \Lambda$, we define the *uncertainty region*

$$\mathcal{U} \;=\; \big\{\, \mathbf{v} \in \mathcal{D} \;\big|\; \forall f \in \Lambda :\; l_f \le f[\mathbf{v}] \le u_f \,\big\}. \tag{4}$$

The region $\mathcal{U}$ contains exactly those parameter instantiations that satisfy all learned interval constraints simultaneously. Together with the pMDP M_Θ, it induces a UMDP $U = (S, A, s_0, \mathcal{P}_\mathcal{U})$, where the uncertainty set consists of all transition kernels obtained by instantiating M_Θ with a parameter vector in $\mathcal{U}$:

$$\mathcal{P}_\mathcal{U} \;:=\; \big\{\, P[\mathbf{v}] \in (\Delta(S))^{S\times A} \;\big|\; \mathbf{v} \in \mathcal{U} \,\big\}.$$

Theorem 1 (Soundness of $\mathcal{U}$). *With probability at least $1-\delta$, the true parameter instantiation $\mathbf{u}$ satisfies $\mathbf{u} \in \mathcal{U}$. Equivalently,*

$$\Pr\big[\, P[\mathbf{u}] \in \mathcal{P}_\mathcal{U} \,\big] \;\ge\; 1-\delta.$$

Consequently, for any policy $\pi \in \Pi$ and state $s \in S$, with probability at least $1-\delta$ the robust value under the induced UMDP lower-bounds the true value in M, i.e.,

$$\Pr\big[\, V_U^\pi(s) \leq V_M^\pi(s) \,\big] \;\geq\; 1-\delta. \qquad \square$$

The induced UMDP U also yields a robust policy π^*. By Theorem 1, π^* achieves, with probability at least $1-\delta$, at least its robust value $V_U^\pi(s)$ on the unknown MDP $M[\mathbf{u}]$. However, computing the robust value and policy from U remains very challenging. We summarise the key issues below and then describe in more detail how we address them in the remainder of this section.

1. **Non-rectangularity and tractability (Sec. 3.1).** The uncertainty set $\mathcal{P}_\mathcal{U}$ for U is in general *non-rectangular*: shared parameters couple uncertainty across states and actions. As a consequence, the robust optimisation problem in (1) becomes very difficult and robust policy synthesis is NP-hard in general [47]. We therefore introduce a family of *rectangular* relaxations that over-approximate $\mathcal{P}_\mathcal{U}$. These relaxations preserve the soundness of Theorem 1, while enabling efficient synthesis. For each relaxation, we analyse (i) the cost of constructing it, (ii) the resulting complexity of robust policy synthesis, and (iii) its tightness. Further, we establish an inclusion hierarchy among the uncertainty sets, making the tightness–efficiency trade-off explicit.
2. **Polynomial constraints and linearisation (Sec. 3.2).** The uncertainty set $\mathcal{P}_\mathcal{U}$ is in general defined by polynomial constraints, since transition probabilities may be polynomial functions of the parameters. As a result, even after rectangularisation, the inner optimisation in (2) can remain challenging, as it requires optimisation under polynomial constraints. We therefore leverage *linear relaxations* that transform these polynomial constraints into linear ones. Combined with rectangular relaxations, we show that this yields tractable inner problems, while retaining tight uncertainty sets.
3. **Possible emptiness of $\mathcal{P}_\mathcal{U}$ (Sec. 3.3).** Finally, we show that, unlike in standard IMDP learning, the induced uncertainty set $\mathcal{P}_\mathcal{U}$ may be empty. We give a statistical interpretation of this edge case, derive an upper bound on the probability that it occurs, and provide a principled fallback mechanism.

3.1 Rectangular Relaxations

Robust policy synthesis over $\mathcal{P}_\mathcal{U}$ is intractable in general, since $\mathcal{P}_\mathcal{U}$ may be non-rectangular. We therefore introduce a range of rectangular relaxations, i.e., uncertainty sets that over-approximate $\mathcal{P}_\mathcal{U}$. These relaxations preserve the inclusion guarantee for the unknown MDP $M[\mathbf{u}]$ while enabling tractable synthesis. For each relaxation, we describe the additional effort required to construct it, as well as the resulting complexity of policy synthesis, measured by the cost of the inner minimisation in (2) in each iteration of robust value iteration. We organise the relaxations into an inclusion hierarchy relative to $\mathcal{P}_\mathcal{U}$ and to the baseline uncertainty set $\mathcal{P}_\mathrm{I}$ induced solely by the intervals, as depicted in Figure 2.

From here on, for ease of presentation, we assume that each parametric expression $f \in \Lambda$ is linear in the parameters. This comes without loss of generality,

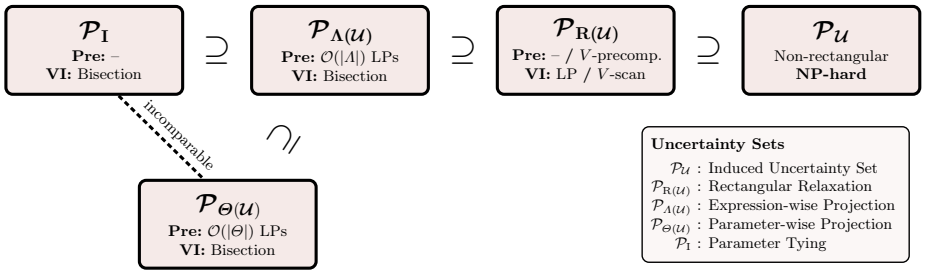


Fig. 2: Inclusion hierarchy of the considered uncertainty sets. For each set, we report the computational effort required to construct the relaxation (**Pre**), in addition to computing the confidence intervals and constructing the model, and the effort per inner minimisation during robust value iteration (**VI**).

as we will discuss in Section 3.2 how to relax polynomial constraints into linear ones while preserving soundness of the resulting uncertainty sets. Recall that we consider the case where the *original* uncertainty sets derived from data are confidence intervals. As we shall argue in Section 3.4, our results on rectangular relaxations extend to other uncertainty classes, such as L_1-balls and ellipsoids.

Rectangularisation via local projections. The most natural way to relax a non-rectangular uncertainty set into a rectangular one is to allow the adversarial environment to choose its worst-case transition probabilities independently in each state–action pair. Formally, given the coupled uncertainty set $\mathcal{P}_{\mathcal{U}}$, its *rectangular relaxation* $\mathcal{P}_{R(\mathcal{U})}$ is the product of its local projections:

$$\mathcal{P}_{R(\mathcal{U})} \;:=\; \prod_{(s,a)\in S\times A} \mathcal{P}_{R(\mathcal{U})}(s,a), \quad \text{where} \quad \mathcal{P}_{R(\mathcal{U})}(s,a) \;:=\; \big\{\, P[\mathbf{v}](s,a) \;\big|\; \mathbf{v}\in\mathcal{U} \,\big\}.$$

Rather than requiring a single instantiation $\mathbf{v}\in\mathcal{U}$ to be used consistently throughout the pMDP, the environment can select an instantiation from $\mathcal{U}$ separately for each state–action pair. Essentially, this corresponds to renaming each shared parameter, thus eliminating dependencies [22,36]. This is a relaxation as it grants the environment more power: it may choose a different worst-case instantiation from $\mathcal{U}$ for each state–action pair, which can only decrease the robust value.

Since each parametric expression $f\in\Lambda$ is linear and constrained by interval bounds $l_f \le f[\mathbf{v}] \le u_f$, the uncertainty region $\mathcal{U}$ is a polytope. Hence, for a state–action pair (s,a), the inner minimisation in (2) reduces to a linear program:

$$\min_{\mathbf{v}\in\mathcal{U}} \sum_{s'\in S} P[\mathbf{v}](s,a,s')\, V^*_U(s').$$

or, equivalently:

$$\min_{\mathbf{v}\in\mathcal{D}} \quad \sum_{s'\in S} P[\mathbf{v}](s,a,s')\, V^*_U(s') \quad \text{s.t.} \quad l_f \le f[\mathbf{v}] \le u_f,\ \forall f\in\Lambda.$$

Hence, policy synthesis via robust value iteration requires solving a linear program for each state–action pair in every iteration. In practice, this can often still

be efficient for two reasons. First, the feasible region is fixed: the constraints defining $\mathcal{U}$ depend only on the parametric expressions and the learned bounds $[l_f, u_f]$, and thus can be constructed once and reused throughout value iteration. Second, successive iterations induce only small changes in the objective, since the coefficients $P[\mathbf{v}](s, a, s')$ are fixed and the value estimates converge. Using a simplex-based LP solver [9,45], we can *warm-start* from the vertex that was optimal in the previous iteration. Since the changes in the objective become smaller as value iteration converges, the previously optimal vertex is often still optimal, or close to optimal, so only few steps are needed to update the solution.

Alternatively, since each inner optimisation is over the same feasible region $\mathcal{U}$, which remains fixed, we can precompute a vertex representation of $\mathcal{U}$, i.e., its set of extreme points. This is closely related to the enumeration approach used in parameter synthesis when the parameter space is a hyperrectangle [22,36]. The inner minimisation then reduces from solving an LP to evaluating the objective at the vertices. However, converting a polytope to vertex representation can be expensive in general, and the number of vertices may grow exponentially with the dimension of the parameter space. This approach is therefore only suitable for low-dimensional parameter spaces, as we examine in the evaluation in Section 4.

Expression-wise projections. We next consider a relaxation that projects $\mathcal{U}$ back to interval bounds over the parametric expressions. This aggregates information from all transitions jointly through $\mathcal{U}$ and can yield substantially tighter intervals than those obtained directly through parameter tying. The resulting model is an IMDP, whose inner optimisation in robust value iteration can be solved efficiently via bisection [23,33].

For each expression $f \in \Lambda$, we derive refined bounds l_f^Λ and u_f^Λ as the minimum and maximum values that $f[\mathbf{v}]$ can attain over $\mathbf{v} \in \mathcal{U}$. Concretely, we solve

$$l_f^\Lambda \;:=\; \min_{\mathbf{v}\in\mathcal{U}} \; f[\mathbf{v}] \qquad\qquad u_f^\Lambda \;:=\; \max_{\mathbf{v}\in\mathcal{U}} \; f[\mathbf{v}].$$

We define the corresponding rectangular interval uncertainty set $\mathcal{P}_{\Lambda(\mathcal{U})}$ as

$$\mathcal{P}_{\Lambda(\mathcal{U})}(s,a) \;:=\; \left\{ \mu \in \Delta(S) \;\middle|\; \forall s' \in S :\; l^\Lambda_{P_\Theta(s,a,s')} \le \mu(s') \le u^\Lambda_{P_\Theta(s,a,s')} \right\}.$$

Computing $\mathcal{P}_{\Lambda(\mathcal{U})}$ requires solving $2|\Lambda|$ linear programs. Robust value iteration on the resulting IMDP then has the same per-iteration cost as the standard interval model, but can yield significantly tighter uncertainty sets since the bounds reflect information aggregated across all transitions through $\mathcal{U}$. Moreover, the individual LPs are independent and can be solved in parallel when constructing the model.

Parameter-wise projections. Expression-wise projections can be expensive when the number of distinct expressions $|\Lambda|$ is large. As an alternative, we project $\mathcal{U}$ to interval constraints on the parameters Θ themselves. Concretely, this yields an axis-aligned over-approximation of the polytope $\mathcal{U}$ by projecting onto each parameter dimension and forming the corresponding enclosing hyperrectangle.

For each parameter $\theta_i \in \Theta$, $1 \le i \le \ell$, we compute its lower and upper bounds over the uncertainty region $\mathcal{U}$ by $\underline{\theta}_i \;:=\; \min_{\mathbf{v}\in\mathcal{U}} \mathbf{v}_i$ and $\overline{\theta}_i \;:=\; \max_{\mathbf{v}\in\mathcal{U}} \mathbf{v}_i$. These

define the smallest hyperrectangle that contains $\mathcal{U}$, $B(\mathcal{U}) := \prod_{i=1}^{\ell}[\underline{\theta}_i, \overline{\theta}_i] \supseteq \mathcal{U}$. We then derive bounds for each expression $f \in \Lambda$ by evaluating it over $B(\mathcal{U})$,

$$l_f^{\Theta} := \min_{\mathbf{v} \in B(\mathcal{U})} f[\mathbf{v}], \qquad u_f^{\Theta} := \max_{\mathbf{v} \in B(\mathcal{U})} f[\mathbf{v}].$$

Since each f is linear, these extrema are attained at a vertex of $B(\mathcal{U})$. Thus, computing l_f^{Θ} and u_f^{Θ} is simpler than the expression-wise projection bounds, as it does not require solving an LP over $\mathcal{U}$, but only selecting, for each parameter θ_i, either $\underline{\theta}_i$ or $\overline{\theta}_i$ according to the sign of its coefficient in f. The resulting interval uncertainty set $\mathcal{P}_{\Theta(\mathcal{U})}$ is then defined analogously to $\mathcal{P}_{\Lambda(\mathcal{U})}$.

Computing $\mathcal{P}_{\Theta(\mathcal{U})}$ requires solving $2|\Theta|$ linear programs over $\mathcal{U}$, which can be significantly cheaper than solving two LPs for every expression in Λ. However, the hyperrectangle $B(\mathcal{U})$ discards dependencies between parameters and may therefore yield considerably looser bounds than expression-wise projection. Moreover, the induced intervals need not be tighter than the original intervals obtained directly from the samples: the two constructions are, in general, incomparable, and either can produce tighter bounds. We provide counterexamples for both directions in Appendix A. To recover inclusion of the original intervals, one can simply intersect the resulting bounds.

We now state the inclusion hierarchy of the discussed uncertainty sets. We denote by $\mathcal{P}_I$ the original interval uncertainty set obtained from the samples using parameter tying as described in Section 2.4.

Theorem 2 (Inclusion hierarchy). *The following inclusions hold:*

$$\mathcal{P}_I \supseteq \mathcal{P}_{\Lambda(\mathcal{U})} \supseteq \mathcal{P}_{R(\mathcal{U})} \supseteq \mathcal{P}_{\mathcal{U}}, \quad \text{and} \quad \mathcal{P}_{\Theta(\mathcal{U})} \supseteq \mathcal{P}_{\Lambda(\mathcal{U})}.$$

Moreover, $\mathcal{P}_{\Theta(\mathcal{U})}$ is in general incomparable to $\mathcal{P}_I$. □

The inclusion hierarchy of Theorem 2 implies that every over-approximating relaxation inherits the same high-confidence inclusion guarantee for the true, unknown MDP as $\mathcal{P}_{\mathcal{U}}$ in Theorem 1. Consequently, all such relaxations can be used for robust policy synthesis with high-confidence performance guarantees.

3.2 Linear Relaxations for Polynomial Constraints

Thus far, we assumed that the parametric expressions in Λ are linear, and hence that $\mathcal{U}$ is a polytope in the parameter space. In general, transition probability expressions in a pMDP may be *polynomials* in the parameters, which makes the projection constraints $l_f \leq f[\mathbf{v}] \leq u_f$ non-linear. As a consequence, the rectangular relaxations introduced in Section 3.1 no longer reduce to linear programs. To retain tractability while preserving soundness, we construct a linear outer approximation of the feasible region using *McCormick envelopes* [31, 39].

McCormick envelopes are linear inequalities that describe the convex hull of a bilinear term $z = xy$ with upper and lower bounds $\underline{x} \leq x \leq \overline{x}$ and $\underline{y} \leq y \leq \overline{y}$ [31]. Concretely, they are given by:

$$\begin{aligned} z &\geq \underline{y}\,x + \underline{x}\,y - \underline{x}\,\underline{y}, & z &\leq \underline{y}\,x + \overline{x}\,y - \overline{x}\,\underline{y}, \\ z &\geq \overline{y}\,x + \overline{x}\,y - \overline{x}\,\overline{y}, & z &\leq \overline{y}\,x + \underline{x}\,y - \underline{x}\,\overline{y}. \end{aligned} \tag{5}$$

While McCormick envelopes provide the exact convex-hull relaxation of a single bilinear term, they can be applied compositionally to obtain a linear outer approximation of an entire polynomial constraint. For instance, for the cubic monomial $\theta_1\theta_2\theta_3$ we introduce auxiliary variables $z_{12} = \theta_1\theta_2$ and $z_{123} = z_{12}\theta_3$ for the intermediate products, and add the McCormick envelopes in (5) for both bilinear terms. Applying this construction to all monomials in the constraint system defining $\mathcal{U}$ in (4) yields a system of linear constraints that over-approximates the original polynomial relations. As an outer approximation, it preserves soundness and can be used within our relaxations. Formal details on the construction and the resulting inequality system, are provided in Appendix D.

Optimisation-based bound tightening. The McCormick inequalities in (5) require finite lower and upper bounds for every variable. We obtain initial bounds by solving two LPs per variable over the current linear relaxation, and then iteratively refine them: tighter bounds yield tighter McCormick envelopes, which in turn can imply further bound improvements. This iterative procedure is known as optimisation-based bound tightening (OBBT) and is a standard domain-reduction technique in global optimisation [20]. In our implementation, we apply OBBT as a preprocessing step to construct a tight linear relaxation of the uncertainty region $\mathcal{U}$. In the experimental evaluation in Section 4, we show that this approach yields tight relaxations and resulting robust guarantees.

3.3 Joint Feasibility and Possible Emptiness of $\mathcal{U}$

Unlike the interval uncertainty sets obtained in standard IMDP learning, the induced uncertainty region $\mathcal{U}$ may be empty. This is a consequence of enforcing *global* consistency with the underlying parametric structure: while each learned interval constraint may be individually feasible, there need not exist a single parameter instantiation that satisfies all of them simultaneously. In that case, the induced uncertainty set $\mathcal{P}_{\mathcal{U}}$ is empty as well, and so are its rectangular relaxations.

If the pMDP M_Θ correctly specifies the underlying system from which the transition samples are obtained, then with probability at least $1-\delta$ the true parameter instantiation $\mathbf{u}$ satisfies all constraints, and hence lies in $\mathcal{U}$. In particular,

$$\Pr[\mathcal{U} \neq \emptyset] \;\geq\; \Pr[\mathbf{u} \in \mathcal{U}] \;\geq\; 1-\delta, \quad \text{or equivalently,} \quad \Pr[\mathcal{U} = \emptyset] \leq \delta.$$

Thus, under correct specification, emptiness can occur with probability at most δ.

Interpretation. If $\mathcal{U}$ is empty, then no admissible parameter instantiation simultaneously satisfies all learned interval constraints. Statistically, this means that the uncertainty statements obtained from the data are jointly incompatible with the assumed parametric structure at confidence level $1-\delta$. Hence, emptiness can be interpreted as evidence against the chosen pMDP model at significance level δ.

This observation is not just an edge case but also an advantage of our approach. By enforcing joint consistency across all learned constraints, it can reveal model misspecification that would be hidden by purely local interval uncertainty sets.

Fallback to interval uncertainty. If $\mathcal{U}$ is empty, one can safely fall back to the original interval uncertainty set learned directly from the samples. In contrast to $\mathcal{U}$, this interval uncertainty set is always non-empty, since for each state-action pair it contains at least the empirical estimate of the corresponding successor distribution, and thus admits at least one feasible distribution per local choice.

3.4 Extension to Other Classes of Uncertainty Sets

We have so far assumed that the base uncertainty sets mapped through the structure of the underlying pMDP into parameter space are intervals. However, our approach and the inclusion hierarchy of Theorem 2 extend beyond intervals. In the following, we discuss two uncertainty classes that illustrate this extension.

L_1 uncertainty sets. One important class is given by L_1-balls [32, 40] based on the Weissman inequality [46]. For each state–action pair, this yields a high-confidence uncertainty set around the empirical estimate $\tilde{P}(s,a)$ in the form of an L_1 ball, with details provided in Appendix E. Since L_1-balls are polytopes, they fit seamlessly with our approach. Concretely, given empirical estimates $\tilde{P}(s,a)$ and radii $\varepsilon(s,a)$ for all $(s,a) \in S \times A$, we define the uncertainty region

$$\mathcal{U}_{L_1} \;=\; \left\{ \mathbf{v} \in \mathcal{D} \;\middle|\; \|P_\Theta[\mathbf{v}](s,a) - \tilde{P}(s,a)\|_1 \leq \varepsilon(s,a) \right\}. \tag{6}$$

Since each $P_\Theta(s,a,s')$ is linear in the parameters, the L_1-constraints can be expressed as linear inequalities. Hence, $\mathcal{U}_{L_1}$ is again a polytope in parameter space, and the rectangular relaxations introduced in Section 3.1 carry over directly.

Ellipsoidal uncertainty sets. Another relevant class is given by *ellipsoids*. These may arise as local uncertainty sets, for instance from second-order approximations of log-likelihood functions [33], or directly in parameter space [4,49]. In both cases, the resulting uncertainty region is generally no longer a polytope. Nevertheless, the rectangular relaxations of Section 3.1, as well as the inclusion hierarchy, carry over.

In particular, for an ellipsoidal parameter region, the local rectangular relaxation can still be obtained by projecting to state–action-wise uncertainty sets, but the corresponding inner optimisation problems are no longer linear programs. Instead, optimising a linear Bellman objective over ellipsoidal constraints yields second-order cone programs (SOCPs) [29]. Likewise, expression-wise projections can be obtained by solving two SOCPs per expression to derive lower and upper bounds, after which robust value iteration proceeds as in the interval case. Details on the construction are provided in Appendix E.

4 Experimental Evaluation

We have integrated our new methods for UMDP learning into the PRISM model checker [27]. We evaluate uncertainty sets from the three relaxations introduced in Section 3: rectangular ($\mathcal{P}_{\mathrm{R}(\mathcal{U})}$), expression-wise ($\mathcal{P}_{\Lambda(\mathcal{U})}$), and parameter-wise ($\mathcal{P}_{\Theta(\mathcal{U})}$) comparing to a baseline of standard interval learning with parameter tying

($\mathcal{P}_\mathrm{I}$). A comparison to ellipsoidal uncertainty sets is provided in Appendix F. We compare both the tightness of the learned uncertainty sets and the computational efficiency of robust solving across a range of established benchmarks with linear and polynomial parameter structures. Table 1 summarises the salient features, detailed description are in Appendix F.

Experimental setup. We evaluate our methods in two settings. First, we compare the tightness of the uncertainty sets and the computational efficiency of constructing and solving the corresponding models from the *same* data, i.e., the same set of collected transition samples C. To this end, we sample a total of 10^5 trajectories from the true model under a uniform sampling policy.

In the second setting, we evaluate the different approaches in an online learning scenario. Here, trajectories are sampled sequentially, the uncertain model is periodically updated from the data collected so far, and the resulting policy is used to guide further exploration. Concretely, we follow the *optimism in the face of uncertainty* principle [3, 19, 43]: whenever the count $\#(s,a)$ doubles for some state–action pair (s,a), we recompute the uncertain model from all trajectories collected so far and update the sampling policy to the *optimistic* policy, i.e., the policy that is optimal for the current uncertain model under the assumption that the environment resolves uncertainty in the agent's favour. This is a standard mechanism for balancing exploration and exploitation in policy learning. We emphasise, however, that our approach is agnostic to the concrete sampling procedure. Once an overall sampling budget is exhausted, we use all collected trajectories to construct the final uncertain model of the respective class and synthesise the corresponding *robust* policy. So, unlike the first setting, the different approaches are not evaluated on a fixed common data set. Instead, the data collection itself is guided by the evolving uncertainty structure, so tighter uncertainty sets can lead to more effective exploration and, ultimately, to improved final policies and guarantees. All uncertain models are built at PAC confidence level $1-\delta = 0.999$.

Table 1: Salient characteristics of the benchmark instances. $\mathbb{P}$ denotes a temporal-logic satisfaction property and $\mathbb{E}$ an expected-reward property.

Benchmark	Instance	|S|	|T|	|Θ|	|Λ|	Prop.
Aircraft	(50, 10)	5463	73033	6	721	$\mathbb{P}$
	(100, 20)	39028	551598	6	1441	$\mathbb{P}$
	(200, 40)	294458	4285228	6	2881	$\mathbb{P}$
Betting Game	(50)	13500	111678	2	2613	$\mathbb{E}$
	(100)	52000	448378	2	5283	$\mathbb{E}$
	(150)	93825	817943	2	7147	$\mathbb{E}$
Engagement	(100)	1994	5962	5	61	$\mathbb{E}$
	(300)	5994	17962	5	61	$\mathbb{E}$
	(1000)	19994	59962	5	61	$\mathbb{E}$
Mars Rover	(50, 50)	191868	1003861	6	643	$\mathbb{P}$
	(75, 75)	550447	2973287	6	943	$\mathbb{P}$
	(125, 125)	1749083	9591771	6	1543	$\mathbb{P}$
Glider	(21, 17)	357	5209	4	777	$\mathbb{E}$
	(51, 47)	2397	37066	4	4977	$\mathbb{E}$
	(106, 99)	4757	74299	4	9768	$\mathbb{E}$
Parallel Betting	(5)	3051	15543	3	477	$\mathbb{E}$
	(10)	21006	187730	3	1077	$\mathbb{E}$
	(20)	151556	1846840	3	185209	$\mathbb{E}$

Results. Table 2 reports the results for the first setting under uniform sampling. For each uncertainty structure, we evaluate how well the resulting UMDP can certify the performance of the policy that is optimal in the true MDP. We report the robust value $\underline{V}$ and the optimistic value $\overline{V}$ of this policy, corresponding to the worst- and best-case value that can be certified under the respective uncertainty structure. This allows a fair comparison of the different approximations. In

Table 2: Comparison of the final bounds and runtimes obtained after sampling 10^5 trajectories uniformly. For each method, $[\underline{V}, \overline{V}]$ denotes the final lower and upper bound on the value of the optimal policy in the true model, and Rel. gap the interval width relative to the true value V^*. TO indicates timeout at 2 [h].

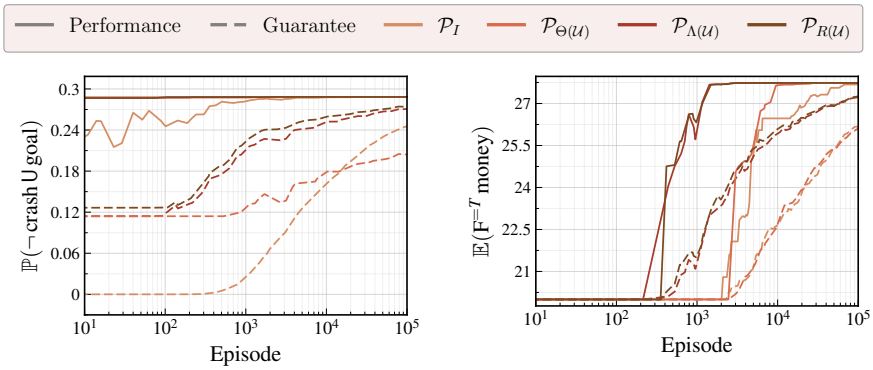
Benchmark	Instance	$\mathcal{P}_I$			$\mathcal{P}_{\Theta(\mathcal{U})}$			$\mathcal{P}_{\Lambda(\mathcal{U})}$			$\mathcal{P}_{R(\mathcal{U})}$		
		$[\underline{V}, \overline{V}]$	Rel. gap	Time [s]	$[\underline{V}, \overline{V}]$	Rel. gap	Time [s]	$[\underline{V}, \overline{V}]$	Rel. gap	Time [s]	$[\underline{V}, \overline{V}]$	Rel. gap	Time [s]
Aircraft	(50, 10)	[0.67, 0.814]	0.19	4.04	[0.618, 0.828]	0.28	4.62	[0.719, 0.767]	0.06	5.72	[0.725, 0.761]	**0.05**	47.12
	(100, 20)	[0.745, 0.906]	0.19	30.45	[0.679, 0.918]	0.29	34.71	[0.805, 0.865]	0.07	39.54	[0.814, 0.858]	**0.05**	353.36
	(200, 40)	[0.628, 0.92]	0.36	688.27	[0.55, 0.928]	0.47	878.04	[0.754, 0.85]	0.12	882.78	[0.769, 0.838]	**0.09**	4464.25
Betting Game	(50)	[14.4, 65.3]	1.88	2.70	[25.8, 28.7]	**0.10**	6.49	[25.8, 28.7]	**0.10**	7.53	[25.8, 28.7]	**0.10**	2.29
	(100)	[16.5, 162]	3.24	13.07	[41.8, 47.7]	**0.13**	12.86	[41.8, 47.7]	**0.13**	26.31	[41.8, 47.7]	**0.13**	17.05
	(150)	[17.5, 263]	3.98	25.18	[56.5, 65.5]	**0.15**	37.98	[56.5, 65.5]	**0.15**	53.16	[56.5, 65.5]	**0.15**	38.34
Engagement	(100)	[38.4, 49.3]	0.26	5.46	[16.6, 5422]	127.35	4.44	[39.8, 45]	**0.12**	1.00	[39.8, 45]	**0.12**	1.82
	(300)	[38.6, 46.8]	0.19	16.55	[18.7, 3615]	84.74	21.30	[40.8, 45.3]	**0.11**	7.95	[40.8, 45.3]	**0.11**	7.27
	(1000)	[39.1, 45.4]	0.15	70.02	[24, 687]	15.63	57.74	[41.3, 43.8]	**0.06**	65.80	[41.3, 43.8]	**0.06**	95.44
Mars Rover	(50, 50)	[0, 0.797]	1.59	92.02	[0.177, 0.833]	1.31	136.87	[0.488, 0.515]	**0.05**	115.15	–	–	TO
	(75, 75)	[0, 0.895]	1.42	521.67	[0.0904, 0.9]	1.28	576.92	[0.619, 0.647]	**0.04**	580.21	–	–	TO
	(125, 125)	[0, 0.877]	1.42	2169.21	[0.0441, 0.922]	1.42	2553.84	[0.599, 0.635]	**0.06**	2306.29	–	–	TO
Glider	(21, 17)	[41.1, 44.6]	0.08	0.25	[42.4, 42.9]	0.01	0.19	[42.5, 42.8]	**0.01**	0.71	[42.5, 42.8]	**0.01**	37.26
	(51, 47)	[62.3, 20001]	293.07	28.92	[58.8, 59.2]	0.01	1.67	[58.8, 59.1]	**0.01**	17.16	[58.8, 59.1]	**0.01**	6268.23
	(106, 99)	[66.4, 35254]	464.04	52.95	[75.6, 76.1]	0.01	3.88	[75.6, 76]	**0.01**	48.50	–	–	TO
Parallel Betting	(5)	[24, 29.4]	0.20	0.31	[25.1, 28.2]	0.12	0.40	[26.1, 27.2]	**0.04**	0.80	[26.1, 27.2]	**0.04**	10.16
	(10)	[26.4, 40.6]	0.44	2.86	[29.7, 34.5]	0.15	3.23	[31.1, 32.9]	0.06	6.42	[31.1, 32.8]	**0.05**	116.52
	(20)	[13, 110]	3.39	21.23	[25.4, 31.7]	**0.22**	666.30	–	–	TO	–	–	TO

addition, we report the *relative gap*, i.e., the width of the interval $[\underline{V}, \overline{V}]$ divided by the true value V^*, as well as the total runtime required to construct and solve the respective uncertain models. Appendix F extends the results, including a breakdown of the runtime spent on model construction and solving.

Figure 3 illustrates the online policy learning process for two representative benchmarks. For each uncertainty structure, we plot the number of processed trajectories against (1) the performance of the robust policy synthesised after that number of trajectories, evaluated in the true (hidden) MDP, shown as solid lines, and (2) the corresponding PAC performance guarantee, shown as dashed lines.

Discussion. In the first setting, where all uncertainty sets are constructed from the same data, the results confirm the behaviour predicted by the inclusion hierarchy of Theorem 2. As expected, the rectangular relaxation $\mathcal{P}_{R(\mathcal{U})}$ of the uncertainty region $\mathcal{U}$ yields the tightest approximation and, consequently, the sharpest certification of policy performance, but at the highest computational cost. In low-dimensional domains such as the Betting Game, $\mathcal{P}_{R(\mathcal{U})}$ remains practically efficient via vertex enumeration, but this becomes ineffective in higher dimensions.

Further, the results show that the expression-wise projection $\mathcal{P}_{\Lambda(\mathcal{U})}$ remains tight while retaining computational efficiency. In many cases, it achieves exactly the same values as $\mathcal{P}_{R(\mathcal{U})}$, but at a cost comparable to the substantially less tight baseline interval uncertainty set $\mathcal{P}_I$ obtained from parameter tying alone. The main limitation arises when the number of distinct expressions $|\Lambda|$ is very large, since constructing $\mathcal{P}_{\Lambda(\mathcal{U})}$ then requires solving a correspondingly large number of LPs. The experiments also illustrate the incomparability of $\mathcal{P}_I$ and $\mathcal{P}_{\Theta(\mathcal{U})}$: the latter can be tighter in some instances at low computational cost, but is overall dominated by the more consistently effective expression-wise projection. Overall, $\mathcal{P}_{\Lambda(\mathcal{U})}$ appears to offer the best balance between computational efficiency and



(a) Aircraft (b) Parallel Betting

Fig. 3: Robust policy learning progress in the online setting. Solid lines show the performance of the current robust policy on the true model. Dashed lines show the corresponding PAC performance guarantee certified for the policy.

approximation tightness. By leveraging the parametric structure, it achieves substantially tighter approximations than the baseline $\mathcal{P}_I$ based solely on parameter tying, in several cases tightening the bounds by orders of magnitude.

We observe a similar pattern in the second setting, where robust policies are learned online from iteratively collected data and progressively refined uncertainty models. Learning with $\mathcal{P}_{R(\mathcal{U})}$ consistently yields the tightest and strongest performance guarantees throughout the learning process. In particular, it improves sample efficiency, measured by the certified performance guarantee achieved after a given number of processed trajectories, by up to an order of magnitude compared to the baseline approach based on $\mathcal{P}_I$. The expression-wise projection $\mathcal{P}_{\Lambda(\mathcal{U})}$ again closely tracks the guarantees obtained via $\mathcal{P}_{R(\mathcal{U})}$, while the results also empirically illustrate the incomparability of $\mathcal{P}_{\Theta(\mathcal{U})}$ and $\mathcal{P}_I$.

The evaluation demonstrates that exploiting the parametric structure can lead to substantial improvements in both the tightness of the learned uncertainty sets and the quality of the resulting robust policies and their certified performance guarantees, in several cases by orders of magnitude. Overall, this highlights the central benefit of our approach: by lifting statistical uncertainty from the transitions to the parameter space and reasoning about it jointly through the pMDP structure, we obtain substantially less conservative robust learning and synthesis than interval-based uncertain MDP learning alone, while maintaining formal PAC guarantees on performance and robustness.

5 Conclusions

We have presented a framework for robust parameter learning in pMDPs that lifts statistical uncertainty from transitions into parameter space. This yields tighter uncertainty sets than traditional UMDP learning while retaining PAC guarantees, and admits tractable relaxations for robust policy synthesis. Our experiments show substantially improved certified guarantees and more effective robust policies.

Acknowledgements

This work was supported in part by the UKRI Erlangen AI Hub on Mathematical and Computational Foundations of AI (No. EP/Y028872/1).

References

1. Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *NIPS*, pages 2312–2320, 2011.
2. Pranav Ashok, Jan Kretínský, and Maximilian Weininger. PAC statistical model checking for Markov decision processes and stochastic games. In *CAV (1)*, volume 11561 of *Lecture Notes in Computer Science*, pages 497–519. Springer, 2019.
3. Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. In *NIPS*, pages 89–96. Curran Associates, Inc., 2008.
4. Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 463–474. PMLR, 2020.
5. Thom Badings, Sebastian Junges, Ahmadreza Marandi, Ufuk Topcu, and Nils Jansen. Efficient sensitivity analysis for parametric robust Markov chains. In *CAV (3)*, Lecture Notes in Computer Science, pages 62–85. Springer, 2023.
6. Thom S. Badings, Alessandro Abate, Nils Jansen, David Parker, Hasan A. Poonawala, and Mariëlle Stoelinga. Sampling-based robust control of autonomous systems with non-Gaussian noise. In *AAAI*, pages 9669–9678. AAAI Press, 2022.
7. Ezio Bartocci, Radu Grosu, Panagiotis Katsaros, C. R. Ramakrishnan, and Scott A. Smolka. Model repair for probabilistic systems. In *TACAS*, Lecture Notes in Computer Science, pages 326–340. Springer, 2011.
8. Nicole Bäuerle and Jonathan Ott. Markov decision processes with average-value-at-risk criteria. *Math. Methods Oper. Res.*, 74(3):361–379, 2011.
9. Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*, volume 6 of *Athena scientific optimization and computation series*. Athena Scientific, 1997.
10. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In P. Thiagarajan, editor, *Proc. 15th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.
11. Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.
12. Radu Calinescu, Carlo Ghezzi, Kenneth Johnson, Mauro Pezzè, Yasmin Rafiq, and Giordano Tamburrelli. Formal verification with confidence intervals to establish quality of service properties of software systems. *IEEE Trans. Reliab.*, 65(1):107–125, 2016.
13. Milan Ceska, Frits Dannenberg, Nicola Paoletti, Marta Kwiatkowska, and Lubos Brim. Precise parameter synthesis for stochastic biochemical systems. *Acta Informatica*, 54(6):589–623, 2017.
14. C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.
15. Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. Planning with hidden parameter polynomial MDPs. In *AAAI*, pages 11963–11971. AAAI Press, 2023.

16. Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, Ivan Papusha, Hasan A. Poonawala, and Ufuk Topcu. Sequential convex programming for the efficient verification of parametric mdps. In *TACAS (2)*, Lecture Notes in Computer Science, pages 133–150, 2017.
17. Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. Convex optimization for parameter synthesis in MDPs. *IEEE Trans. Autom. Control.*, 67(12):6333–6348, 2022.
18. Kasper Engelen, Guillermo A. Pérez, and Marnix Suilen. Data-efficient safe policy improvement using parametric structure. *CoRR*, abs/2507.15532, 2025.
19. Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Emma Brunskill. Regret minimization in MDPs with options without prior knowledge. In *NIPS*, pages 3166–3176, 2017.
20. Ambros M. Gleixner, Timo Berthold, Benjamin Müller, and Stefan Weltge. Three enhancements for optimization-based bound tightening. *J. Glob. Optim.*, 67(4):731–757, 2017.
21. Ernst Moritz Hahn, Tingting Han, and Lijun Zhang. Synthesis for PCTL in parametric Markov decision processes. In *NASA Formal Methods*, Lecture Notes in Computer Science, pages 146–161. Springer, 2011.
22. Linus Heck, Tim Quatmann, Jip Spel, Joost-Pieter Katoen, and Sebastian Junges. Generalized parameter lifting: Finer abstractions for parametric markov chains. In *ATVA*, Lecture Notes in Computer Science, pages 207–230. Springer, 2025.
23. Garud N. Iyengar. Robust dynamic programming. *Math. Oper. Res.*, 30(2):257–280, 2005.
24. Sebastian Junges, Erika Ábrahám, Christian Hensel, Nils Jansen, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. Parameter synthesis for Markov models: covering the parameter space. *Formal Methods Syst. Des.*, 62(1):181–259, 2024.
25. Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-constrained reinforcement learning for MDPs. In *TACAS*, volume 9636 of *Lecture Notes in Computer Science*, pages 130–146. Springer, 2016.
26. Mykel Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. 2015.
27. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.
28. Ruggero Lanotte, Andrea Maggiolo-Schettini, and Angelo Troina. Parametric probabilistic transition systems for system design and analysis. *Formal Aspects Comput.*, 19(1):93–109, 2007.
29. Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra Appl.*, 284(1–3):193–228, 1998.
30. Fan Long and Martin C. Rinard. Automatic patch generation by learning correct code. In *POPL*, pages 298–312. ACM, 2016.
31. Garth P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I - convex underestimating problems. *Math. Program.*, 10(1):147–175, 1976.
32. Tobias Meggendorfer, Maximilian Weininger, and Patrick Wienhöft. What are the odds? improving statistical model checking of Markov decision processes. In *Proc. 2nd International Joint Conference on Quantitative Evaluation of Systems and Formal Modeling and Analysis of Timed Systems (QEST+FORMATS'25)*, LNCS, pages 195–218. Springer, 2025.

33. Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Oper. Res.*, 53(5):780–798, 2005.
34. Elizabeth Polgreen, Viraj B. Wijesuriya, Sofie Haesaert, and Alessandro Abate. Data-efficient Bayesian verification of parametric Markov chains. In *QEST*, volume 9826 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2016.
35. Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.
36. Tim Quatmann, Christian Dehnert, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Parameter synthesis for Markov models: Faster than ever. In *ATVA*, volume 9938 of *Lecture Notes in Computer Science*, pages 50–67, 2016.
37. Yannik Schnitzer, Alessandro Abate, and David Parker. Certifiably robust policies for uncertain parametric environments. In *TACAS (3)*, volume 15698 of *Lecture Notes in Computer Science*, pages 63–83. Springer, 2025.
38. Yannik Schnitzer, Alessandro Abate, and David Parker. Efficient solution and learning of robust factored mdps. *AAAI*, 2026.
39. Joseph K. Scott, Matthew D. Stuber, and Paul I. Barton. Generalized mccormick relaxations. *J. Glob. Optim.*, 51(4):569–606, 2011.
40. Alexander L. Strehl and Michael L. Littman. A theoretical analysis of model-based interval estimation. In *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 856–863. ACM, 2005.
41. Jörg Stückler, Max Schwarz, Mark Schadler, Angeliki Topalidou-Kyniazopoulou, and Sven Behnke. Nimbro explorer: Semiautonomous exploration and mobile manipulation in rough terrain. *J. Field Robotics*, 33(4):411–430, 2016.
42. Marnix Suilen, Thom S. Badings, Eline M. Bovy, David Parker, and Nils Jansen. Robust markov decision processes: A place where AI and formal methods meet. In *Principles of Verification (3)*, volume 15262 of *Lecture Notes in Computer Science*, pages 126–154. Springer, 2024.
43. Marnix Suilen, Thiago D. Simão, David Parker, and Nils Jansen. Robust anytime learning of Markov decision processes. In *NeurIPS*, 2022.
44. Leslie G. Valiant. A theory of the learnable. In *STOC*, pages 436–445. ACM, 1984.
45. Robert J. Vanderbei. *Linear programming - foundations and extensions*, volume 4 of *Kluwer international series in operations research and management service*. Kluwer, 1998.
46. Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdú, and Marcelo J. Weinberger. Inequalities for the l_1 deviation of the empirical distribution. Technical Report HPL-2003-97(R.1), Hewlett-Packard Laboratories, 2003.
47. Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Math. Oper. Res.*, 38(1):153–183, 2013.
48. Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Robust control of uncertain markov decision processes with temporal logic specifications. In *CDC*. IEEE, 2012.
49. Dongruo Zhou, Quanquan Gu, and Csaba Szepesvári. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In *Proceedings of the 34th Annual Conference on Learning Theory*, volume 134, pages 4532–4576. PMLR, 2021.

A Proofs

We provide the formal proofs for the Theorems 1 and 2.

A.1 Proof of Theorem 1

Theorem 1 (Soundness of $\mathcal{U}$, restated)**.** *With probability at least* $1-\delta$*, the true parameter instantiation* $\mathbf{u}$ *satisfies* $\mathbf{u} \in \mathcal{U}$*. Equivalently,*

$$\Pr\big[\, P[\mathbf{u}] \in \mathcal{P}_{\mathcal{U}} \,\big] \;\geq\; 1-\delta.$$

Consequently, for any policy $\pi \in \Pi$ *and state* $s \in S$*, with probability at least* $1-\delta$ *the robust value under the induced UMDP lower-bounds the true value in* M*, i.e.,*

$$\Pr\big[\, V_U^{\pi}(s) \leq V_M^{\pi}(s) \,\big] \;\geq\; 1-\delta.$$

Proof. For each non-constant expression $f \in \Lambda_{\mathsf{unk}}$, the learned interval satisfies

$$\Pr\big[\, l_f \leq f[\mathbf{u}] \leq u_f \,\big] \;\geq\; 1 - \frac{\delta}{|\Lambda_{\mathsf{unk}}|}.$$

For every constant expression $f \in \Lambda \setminus \Lambda_{\mathsf{unk}}$, we set $l_f = u_f = f$, and hence $l_f \leq f[\mathbf{u}] \leq u_f$ holds deterministically.y Now consider the event

$$E \;:=\; \big\{\, \forall f \in \Lambda :\; l_f \leq f[\mathbf{u}] \leq u_f \,\big\}.$$

If E does not occur, then at least one of the intervals for some $f \in \Lambda_{\mathsf{unk}}$ fails to contain the true value $f[\mathbf{u}]$. Thus

$$E^c \;\subseteq\; \bigcup_{f \in \Lambda_{\mathsf{unk}}} \big\{\, f[\mathbf{u}] \notin [l_f, u_f] \,\big\},$$

and by the union bound,

$$\Pr(E^c) \;\leq\; \sum_{f \in \Lambda_{\mathsf{unk}}} \Pr\big[\, f[\mathbf{u}] \notin [l_f, u_f] \,\big] \;\leq\; \sum_{f \in \Lambda_{\mathsf{unk}}} \frac{\delta}{|\Lambda_{\mathsf{unk}}|} \;=\; \delta.$$

Hence $\Pr(E) \geq 1-\delta$. By definition of $\mathcal{U}$, the event E is exactly the event $\mathbf{u} \in \mathcal{U}$.

For the second claim, note that by definition of $\mathcal{P}_{\mathcal{U}}$, we have $P[\mathbf{u}] \in \mathcal{P}_{\mathcal{U}}$ if and only if $\mathbf{u} \in \mathcal{U}$. Therefore,

$$\Pr\big[\, P[\mathbf{u}] \in \mathcal{P}_{\mathcal{U}} \,\big] = \Pr\big[\, \mathbf{u} \in \mathcal{U} \,\big] \;\geq\; 1-\delta.$$

For the third claim, fix any policy $\pi \in \Pi$ and state $s \in S$. On the event $P[\mathbf{u}] \in \mathcal{P}_{\mathcal{U}}$, the true kernel is admissible, and therefore

$$V_U^{\pi}(s) = \inf_{P' \in \mathcal{P}_{\mathcal{U}}} V_{P'}^{\pi}(s) \;\leq\; V_{P[\mathbf{u}]}^{\pi}(s) = V_M^{\pi}(s).$$

Hence the event $\{P[\mathbf{u}] \in \mathcal{P}_{\mathcal{U}}\}$ implies the event $\{V_U^{\pi}(s) \leq V_M^{\pi}(s)\}$, and so

$$\Pr\big[\, V_U^{\pi}(s) \leq V_M^{\pi}(s) \,\big] \;\geq\; \Pr\big[\, P[\mathbf{u}] \in \mathcal{P}_{\mathcal{U}} \,\big] \;\geq\; 1-\delta.$$

□

A.2 Proof of Theorem 2

Theorem 2 (Inclusion hierarchy, restated)**.** *The following inclusions hold:*

$$\mathcal{P}_I \supseteq \mathcal{P}_{\Lambda(\mathcal{U})} \supseteq \mathcal{P}_{R(\mathcal{U})} \supseteq \mathcal{P}_{\mathcal{U}}, \qquad \text{and} \qquad \mathcal{P}_{\Theta(\mathcal{U})} \supseteq \mathcal{P}_{\Lambda(\mathcal{U})}.$$

Moreover, $\mathcal{P}_{\Theta(\mathcal{U})}$ is in general incomparable to $\mathcal{P}_I$.

Proof. We prove each inclusion individually.

1. $\mathcal{P}_{R(\mathcal{U})} \supseteq \mathcal{P}_{\mathcal{U}}$. Let $P[\mathbf{v}] \in \mathcal{P}_{\mathcal{U}}$ for some $\mathbf{v} \in \mathcal{U}$. Then, by definition of the local projections, we have $P[\mathbf{v}](s,a) \in \mathcal{P}_{R(\mathcal{U})}(s,a)$ for every $(s,a) \in S \times A$. Hence $P[\mathbf{v}] \in \mathcal{P}_{R(\mathcal{U})}$, which shows $\mathcal{P}_{\mathcal{U}} \subseteq \mathcal{P}_{R(\mathcal{U})}$.

2. $\mathcal{P}_{\Lambda(\mathcal{U})} \supseteq \mathcal{P}_{R(\mathcal{U})}$. Fix a state–action pair (s,a) and let $\mu \in \mathcal{P}_{R(\mathcal{U})}(s,a)$. By definition, there exists $\mathbf{v} \in \mathcal{U}$ such that $\mu = P[\mathbf{v}](s,a)$. For each successor $s' \in S$, let $f = P_\Theta(s,a,s') \in \Lambda$. Since l_f^Λ and u_f^Λ are defined as the minimum and maximum of $f[\mathbf{v}']$ over all $\mathbf{v}' \in \mathcal{U}$, we have $l_f^\Lambda \le f[\mathbf{v}] = \mu(s') \le u_f^\Lambda$. Thus $\mu \in \mathcal{P}_{\Lambda(\mathcal{U})}(s,a)$. Since (s,a) was arbitrary, it follows that $\mathcal{P}_{R(\mathcal{U})} \subseteq \mathcal{P}_{\Lambda(\mathcal{U})}$.

3. $\mathcal{P}_I \supseteq \mathcal{P}_{\Lambda(\mathcal{U})}$. Recall that $\mathcal{U}$ is defined by the original learned bounds, i.e.,

$$\mathcal{U} = \left\{ \mathbf{v} \in \mathcal{D} \mid \forall f \in \Lambda :\ l_f \le f[\mathbf{v}] \le u_f \right\}.$$

Hence, for every $f \in \Lambda$, all values of $f[\mathbf{v}]$ with $\mathbf{v} \in \mathcal{U}$ lie in the interval $[l_f, u_f]$. Since $l_f^\Lambda = \min_{\mathbf{v}\in\mathcal{U}} f[\mathbf{v}]$ and $u_f^\Lambda = \max_{\mathbf{v}\in\mathcal{U}} f[\mathbf{v}]$, it follows that

$$l_f \le l_f^\Lambda \le u_f^\Lambda \le u_f.$$

Therefore, for every state–action pair (s,a) and successor $s' \in S$, the interval induced by expression-wise projection is contained in the original interval. Thus $\mathcal{P}_{\Lambda(\mathcal{U})}(s,a) \subseteq \mathcal{P}_I(s,a)$ for all (s,a), and consequently $\mathcal{P}_{\Lambda(\mathcal{U})} \subseteq \mathcal{P}_I$.

4. $\mathcal{P}_{\Theta(\mathcal{U})} \supseteq \mathcal{P}_{\Lambda(\mathcal{U})}$. Since $B(\mathcal{U})$ is the smallest axis-aligned hyperrectangle containing $\mathcal{U}$, we have $\mathcal{U} \subseteq B(\mathcal{U})$. Therefore, for every expression $f \in \Lambda$, minimizing or maximizing over the larger set $B(\mathcal{U})$ can only decrease the lower bound and increase the upper bound. In other words, $l_f^\Theta \le l_f^\Lambda$ and $u_f^\Theta \ge u_f^\Lambda$. Thus the interval induced by parameter-wise projection contains the corresponding interval induced by expression-wise projection, and so $\mathcal{P}_{\Lambda(\mathcal{U})}(s,a) \subseteq \mathcal{P}_{\Theta(\mathcal{U})}(s,a)$ for all (s,a). Hence $\mathcal{P}_{\Lambda(\mathcal{U})} \subseteq \mathcal{P}_{\Theta(\mathcal{U})}$.

5. Incomparability of $\mathcal{P}_{\Theta(\mathcal{U})}$ and $\mathcal{P}_I$. We give two counterexamples showing that neither inclusion holds in general.

$\mathcal{P}_{\Theta(\mathcal{U})} \not\subseteq \mathcal{P}_I$. Consider two parameters $\theta_1, \theta_2 \in [0,1]$ and three transition expressions

$$f_1(\theta) = \theta_1, \qquad f_2(\theta) = \theta_2, \qquad f_3(\theta) = \tfrac{1}{2}(\theta_1 + \theta_2),$$

arising from distinct state–action pairs. Suppose the learned intervals are $f_1 \in [0,1]$, $f_2 \in [0,1]$, and $f_3 \in [\frac{1}{2}, \frac{1}{2}]$. Then the corresponding uncertainty region is

$$\mathcal{U} = \{\, (\theta_1, \theta_2) \in [0,1]^2 \mid \theta_1 + \theta_2 = 1 \,\},$$

and thus $B(\mathcal{U}) = [0,1]^2$. For the expression f_3, expression-wise projection yields the exact interval $[l^{\Lambda}_{f_3}, u^{\Lambda}_{f_3}] = [\frac{1}{2}, \frac{1}{2}]$, whereas parameter-wise projection over $B(\mathcal{U})$ yields $[l^{\Theta}_{f_3}, u^{\Theta}_{f_3}] = [0,1]$. Hence $\mathcal{P}_{\Theta(\mathcal{U})}$ admits kernels whose f_3-component lies outside the original interval, showing that $\mathcal{P}_{\Theta(\mathcal{U})} \not\subseteq \mathcal{P}_I$.

$\mathcal{P}_I \not\subseteq \mathcal{P}_{\Theta(\mathcal{U})}$. Consider now a pMDP with one parameter $\theta \in [0,1]$ and two distinct state–action pairs with transition expressions $f_1(\theta) = \theta$ and $f_2(\theta) = 1-\theta$. Suppose the learned intervals are $f_1 \in [0.4, 0.7]$ and $f_2 \in [0.4, 0.7]$. Then

$$\mathcal{U} = \{\, \theta \in [0,1] \mid 0.4 \le \theta \le 0.7,\ 0.4 \le 1-\theta \le 0.7 \,\} = [0.4, 0.6].$$

Thus $B(\mathcal{U}) = [0.4, 0.6]$, and parameter-wise projection yields the refined intervals $f_1, f_2 \in [0.4, 0.6]$. In contrast, the original interval uncertainty set $\mathcal{P}_I$ still permits the independent choices $f_1 = 0.7$ and $f_2 = 0.7$ at the two distinct state–action pairs. Such a kernel belongs to $\mathcal{P}_I$ but not to $\mathcal{P}_{\Theta(\mathcal{U})}$. Hence $\mathcal{P}_I \not\subseteq \mathcal{P}_{\Theta(\mathcal{U})}$. □

B Correspondence between pMDPs and UMDPs

We formalise the relationship between parametric MDPs and uncertain MDPs.

B.1 From pMDPs to UMDPs

Let $M_\Theta = (S, A, s_0, \Theta, P_\Theta)$ be a pMDP and let $\mathcal{U} \subseteq \mathbb{R}^{|\Theta|}$ be a set of admissible parameter instantiations. Each $\mathbf{u} \in \mathcal{U}$ induces a concrete transition kernel $P[\mathbf{u}] \in (\Delta(S))^{S \times A}$. We define the UMDP induced by M_Θ and $\mathcal{U}$ as

$$U_{\mathcal{U}} \coloneqq (S, A, s_0, \mathcal{P}_{\mathcal{U}}),$$

where the uncertainty set is given by

$$\mathcal{P}_{\mathcal{U}} \;\coloneqq\; \{\, P[\mathbf{u}] \in (\Delta(S))^{S \times A} \mid \mathbf{u} \in \mathcal{U} \,\}.$$

The structure of the uncertainty set $\mathcal{P}_{\mathcal{U}}$ is determined by the parameter dependencies in the transition function P_Θ and by the shape of the admissible instantiation set $\mathcal{U}$. In particular, shared parameters induce coupling between transition probabilities at different state–action pairs.

Coupled uncertainty. If there exists a parameter $\theta \in \Theta$ that occurs in the transition polynomials of two distinct state–action pairs (s, a) and (s', a'), then instantiating v simultaneously constrains both $P(s, a)$ and $P(s', a')$. In this case, the resulting uncertainty set $\mathcal{P}_{\mathcal{U}}$ is, in general, non-rectangular.

Rectangular uncertainty. Suppose that the parameter set Θ can be partitioned as $\Theta = \biguplus_{(s,a)\in S\times A} \Theta_{s,a}$ such that $P(s,a)$ depends only on parameters in $\Theta_{s,a}$, and that the admissible instantiation set factorises as

$$\mathcal{U} = \prod_{(s,a)\in S\times A} \mathcal{U}_{s,a}, \qquad \mathcal{U}_{s,a} \subseteq \mathbb{R}^{|\Theta_{s,a}|}.$$

Then the induced uncertainty set is (s,a)-rectangular and can be written as

$$\mathcal{P}_{\mathcal{U}} = \prod_{(s,a)\in S\times A} \{\, P(s,a)[\mathbf{u}_{s,a}] \mid \mathbf{u}_{s,a} \in \mathcal{U}_{s,a} \,\},$$

where $\mathbf{u}_{s,a}$ denotes the restriction of $\mathbf{u}$ to $\Theta_{s,a}$.

B.2 From UMDPs to pMDPs

Conversely, any UMDP can be represented as a pMDP together with a suitable admissible parameter set. Let $U = (S, A, s_0, \mathcal{P})$ be a UMDP, where $\mathcal{P} \subseteq (\Delta(S))^{S\times A}$ is a non-empty set of admissible transition kernels.

Rectangular case. Assume that the uncertainty set $\mathcal{P}$ is (s,a)-rectangular, i.e.,

$$\mathcal{P} = \prod_{(s,a)\in S\times A} \mathcal{P}(s,a) \qquad \text{with} \qquad \mathcal{P}(s,a) \subseteq \Delta(S).$$

We construct an equivalent pMDP by introducing explicit parameters for the transition probabilities. For each $(s,a) \in S \times A$ and each $s' \in S$, let $\theta_{s,a,s'}$ be a parameter representing the probability of transitioning from (s,a) to s'. Let

$$\Theta \coloneqq \{\, \theta_{s,a,s'} \mid (s,a) \in S \times A,\ s' \in S \,\}.$$

Define the parametric transition function P_Θ by

$$P_\Theta(s,a)(s') \coloneqq \theta_{s,a,s'}.$$

The admissible parameter instantiations are restricted to

$$\mathcal{U} \coloneqq \Big\{ \mathbf{u} \in \mathbb{R}^{|\Theta|} \;\Big|\; \big(u(\theta_{s,a,s'})\big)_{s'\in S} \in \mathcal{P}(s,a) \text{ for all } (s,a) \in S \times A \Big\}.$$

By construction, each $\mathbf{u} \in \mathcal{U}$ induces a transition kernel $P[\mathbf{u}] \in \mathcal{P}$, and every kernel in $\mathcal{P}$ arises from a unique instantiation $\mathbf{u} \in \mathcal{U}$. Hence, the pMDP together with $\mathcal{U}$ induces exactly the UMDP U.

Coupled case. Assume that the uncertainty set $\mathcal{P} \subseteq (\Delta(S))^{S\times A}$ is non-rectangular. We again construct a pMDP by introducing explicit parameters for the transition probabilities, but without imposing any independence assumptions.

For each $(s,a) \in S \times A$ and $s' \in S$, let $\theta_{s,a,s'}$ be a parameter representing the probability of transitioning from (s,a) to s'. Let

$$\Theta \coloneqq \{\, \theta_{s,a,s'} \mid (s,a) \in S \times A,\ s' \in S \,\}.$$

Define the parametric transition function P by

$$P_\Theta(s,a)(s') \coloneqq \theta_{s,a,s'}.$$

The admissible parameter instantiations are restricted to

$$\mathcal{U} \;\coloneqq\; \Big\{ \mathbf{u} \in \mathbb{R}^{|\Theta|} \;\Big|\; P[\mathbf{u}] \in \mathcal{P} \Big\}.$$

By construction, $\mathcal{U}$ encodes all global constraints and dependencies between transition probabilities captured by $\mathcal{P}$. Moreover, the pMDP together with $\mathcal{U}$ induces exactly the uncertainty set $\mathcal{P}$, and is non-rectangular whenever $\mathcal{P}$ is.

Together, the constructions show that pMDPs and UMDPs are equivalent modelling formalisms when paired with a suitable admissible parameter set.

C Parameter Tying

This appendix gives the formal details of the parameter-tying preprocessing step used in Section 2.4. The key idea is that transitions labelled by the same parametric expression correspond to the same unknown probability under any parameter instantiation and can therefore be learned from pooled counts. We assume throughout that no state–action distribution assigns the same parametric expression to two distinct successors, i.e.,

$$P_\Theta(s,a,s') = P_\Theta(s,a,s'') \quad\Longrightarrow\quad s' = s''.$$

This assumption is made only for notational and technical convenience. If the same expression labels two distinct successors of a fixed state–action pair, then these occurrences do not give rise to separate Bernoulli events, but are coupled as part of the same successor distribution. Such cases can be eliminated without loss of generality: if two distinct successors s', s'' of the same state–action pair carry the same expression, we introduce an intermediate state that is reached with the shared probability, followed by a fixed distribution over the original successors. This preserves the relevant parametric structure and probabilities while ensuring that each expression labels at most one successor per state–action pair [18].

For an expression $f \in \Lambda$, we define its occurrence set by

$$\mathrm{Occ}(f) \;\coloneqq\; \big\{\, (s,a,s') \in S \times A \times S \mid P_\Theta(s,a,s') = f \,\big\}.$$

All transitions in $\mathrm{Occ}(f)$ share the same parametric expression and hence the true probability value $f[\mathbf{u}]$. Consequently, they correspond to repeated observations of the same Bernoulli event and their counts can be pooled. We therefore define the pooled trial count and pooled success count for f by

$$N_f \;\coloneqq \sum_{(s,a,s')\in \mathrm{Occ}(f)} \#(s,a), \qquad K_f \;\coloneqq \sum_{(s,a,s')\in \mathrm{Occ}(f)} \#(s,a,s').$$

Under the true instantiation $\mathbf{u}$, the random variable K_f is binomially distributed with parameters N_f and $f[\mathbf{u}]$. Hence, any binomial confidence-interval construction from Section 2.3 can be applied directly to the pooled counts (K_f, N_f). We distribute the confidence budget only across the non-constant expressions,

$$\Lambda_{\mathsf{unk}} \;:=\; \{\, f \in \Lambda \mid f \notin \mathbb{Q} \,\}, \qquad \gamma \;:=\; \frac{\delta}{|\Lambda_{\mathsf{unk}}|}.$$

For each $f \in \Lambda_{\mathsf{unk}}$, we compute from a confidence interval $[l_f, u_f]$ such that

$$\Pr\big[\, l_f \le f[\mathbf{u}] \le u_f \,\big] \;\ge\; 1 - \gamma.$$

For constant expressions $f \in \Lambda \setminus \Lambda_{\mathsf{unk}}$, we set $l_f = u_f = f$ exactly. The resulting tied interval uncertainty set is obtained by assigning to each transition (s, a, s') the interval associated with its expression:

$$I(s, a, s') \;:=\; \big[l_{P_\Theta(s,a,s')},\, u_{P_\Theta(s,a,s')}\big].$$

Equivalently, the parameter-tied IMDP is the uncertainty set

$$\mathcal{P}_I(s, a) \;:=\; \Big\{ \mu \in \Delta(S) \;\Big|\; \forall s' \in S :\; \mu(s') \in I(s, a, s') \Big\}.$$

Since all occurrences of a fixed expression share the same pooled interval, parameter tying enforces exact equality between transitions labelled by the same expression. Compared to the standard interval construction that learns a separate interval for every transition probability, parameter tying improves statistical efficiency in two ways. First, pooling increases the effective sample size from the local count $\#(s, a)$ to the aggregated count N_f, which typically yields narrower intervals. Second, the confidence budget is split across $|\Lambda_{\mathsf{unk}}|$ unknown expressions rather than across all non-constant transitions, which tightens the intervals further whenever the same expression occurs multiple times.

Parameter tying applies only to *exact* equality of expressions. If two transitions depend on the same parameters but are labelled by different expressions, then their success probabilities need not coincide under the true instantiation, and the corresponding samples do not form repeated observations of the same Bernoulli event. In that case, pooling is not statistically justified. This is precisely the gap addressed by our parameter-space approach in Section 3, which reasons jointly over all learned constraints instead of only over exactly matching expressions.

D Linear Relaxations for Polynomial Constraints

This appendix provides further details on the linear relaxation of polynomial constraints. The goal is to replace the nonlinear constraint system defining $\mathcal{U}$ by a polyhedral outer approximation that preserves soundness and can therefore be used within the rectangular relaxations of Section 3.1.

Rewriting of monomials into bilinear terms. We view each polynomial expression $f \in \Lambda$ as a factorable expression built from constants, parameters,

additions, and products. Since sums are already linear, only products require relaxation. Concretely, every monomial

$$m(\theta) \;=\; c \prod_{j=1}^{k} \theta_{i_j}$$

is rewritten by introducing auxiliary variables for intermediate products. We introduce auxiliary variables $z_1, \ldots, z_{k-1}$ and enforce the chain

$$z_1 = \theta_{i_1}\theta_{i_2}, \qquad z_r = z_{r-1}\theta_{i_{r+1}} \quad \text{for } r = 2, \ldots, k-1,$$

with $m(\theta) = c\, z_{k-1}$. Applying this construction to all monomials in all expressions yields an equivalent lifted representation of the nonlinear system in terms of linear equalities and bilinear equalities.

McCormick relaxation of bilinear terms. Consider a bilinear equality $z = xy$ with bounds $\underline{x} \le x \le \overline{x}$ and $\underline{y} \le y \le \overline{y}$. Its McCormick relaxation is given by the four inequalities

$$\begin{aligned} z &\ge \underline{y}\, x + \underline{x}\, y - \underline{x}\, \underline{y}, & z &\le \underline{y}\, x + \overline{x}\, y - \overline{x}\, \underline{y}, \\ z &\ge \overline{y}\, x + \overline{x}\, y - \overline{x}\, \overline{y}, & z &\le \overline{y}\, x + \underline{x}\, y - \underline{x}\, \overline{y}. \end{aligned} \tag{7}$$

Over the box $[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, these inequalities describe the convex hull of the graph of $z = xy$ [31]. Hence, replacing every bilinear equality in the lifted system by (7) yields a linear outer approximation of the original polynomial constraints.

Global linear relaxation. Let x collect all base parameters and all auxiliary variables introduced for intermediate products. The original nonlinear uncertainty region is defined by the constraints

$$l_f \le f[\mathbf{v}] \le u_f \qquad \text{for all } f \in \Lambda,$$

together with the box constraints $\mathbf{v} \in \mathcal{D}$. After lifting and replacing each bilinear equality by its McCormick envelope, we obtain a polyhedron

$$\mathcal{R} \;=\; \{\, x \mid Ax \le b \,\}, \tag{8}$$

where $Ax \le b$ contains: (i) the original linear constraints, (ii) all McCormick inequalities for bilinear terms, and (iii) the variable bounds used in these inequalities. By construction, every feasible point of the original nonlinear system extends to a feasible point of $\mathcal{R}$. Hence $\mathcal{R}$ is an outer approximation of the true feasible region, and any robust lower bound computed over $\mathcal{R}$ remains sound.

Optimisation-based bound tightening. The quality of the McCormick relaxation depends crucially on the bounds used in (7). We therefore apply optimisation-based bound tightening (OBBT), which iteratively improves these bounds by solving LPs over the current relaxation. Concretely, for every variable w in the lifted system, including both base parameters and auxiliary variables, we compute

$$\underline{w}^{\,*} \;=\; \min\{\, w \mid Ax \le b \,\}, \qquad \overline{w}^{\,*} \;=\; \max\{\, w \mid Ax \le b \,\}.$$

These optimised bounds replace the previous interval bounds of w, after which all McCormick inequalities involving w are rebuilt. This process is repeated until no bound improves beyond a prescribed tolerance or a maximum number of rounds.

Since tighter variable bounds yield tighter McCormick envelopes, OBBT can significantly strengthen the final linear relaxation. Soundness is preserved throughout, since each OBBT step optimises over the current outer approximation and therefore only removes values that are infeasible in that approximation.

Remarks on tightness. McCormick relaxations are exact for a single bilinear term on a box, but for general factorable polynomials the resulting polyhedral relaxation depends on the chosen factorisation and on the variable bounds. In particular, different recursive decompositions of multilinear monomials can lead to different relaxations, and tighter bounds from OBBT typically improve the quality of the final relaxation substantially. This is well known in deterministic global optimisation and is one of the main motivations for combining McCormick relaxations with OBBT in practice [20, 31, 39].

In our implementation, the construction of $\mathcal{R}$ and the OBBT procedure are performed once as a preprocessing step. The resulting polyhedral feasible region is then reused across all states and throughout robust value iteration, exactly as in the linear case considered in Section 3.1.

E Ellipsoidal Confidence Sets in Parameter Space

For completeness, we also include an ellipsoidal baseline in parameter space. This baseline is obtained by instantiating, in our pMDP setting, the self-normalised least-squares confidence construction of Abbasi-Yadkori et al. [1], together with the linear-MDP adaptation discussed by Ayoub et al. (Appendix C) [4]. In contrast to the polytopic uncertainty sets used in the main body of the paper, this yields an ellipsoidal confidence region over parameter instantiations and therefore leads to conic optimisation in the robust Bellman updates.

Let $\mathcal{M}_\Theta = (S, A, s_0, P)$ be a pMDP with parameter space $D \subseteq \mathbb{R}^\ell$, and let Λ denote the set of distinct parametric transition expressions occurring in P. For each $(s, a, s') \in S \times A \times S$, we write

$$P(s, a, s') = f_{s,a,s'} \in \Lambda.$$

When all transition expressions are affine, each $f \in \Lambda$ can be written as

$$f[\mathbf{u}] = c_f + b_f^\top \mathbf{u},$$

for some constant $c_f \in \mathbb{R}$, coefficients $b_f \in \mathbb{R}^\ell$, and parameter instantiation $\mathbf{u} \in D$.

Ayoub et al. [4] derive confidence ellipsoids for finite-horizon linear MDPs by combining least-squares regression with the self-normalised concentration bound of Abbasi-Yadkori et al. [1]. Their regression target is the overall value itself and tailored to episodic, finite horizon MDPs. In our case, the unknown object is the

transition kernel induced by the unknown parameter instantiation $\mathbf{u}$, for infinite-horizon objectives. To extend this to our setting, we apply the same concentration mechanism directly to the obtained one-step transition observations.

Concretely, for any fixed transition triple (s, a, s'), each visit to (s, a) in the set of observed transitions C contributes a binary observation $Y \in \{0, 1\}$ indicating whether the next state equals s'. Its mean is precisely the transition probability $P[\mathbf{u}](s, a, s')$. Hence the centred noise term

$$\eta := Y - P[\mathbf{u}](s, a, s')$$

is supported on an interval of length 1, and is therefore $\frac{1}{2}$-sub-Gaussian by Hoeffding's lemma [11]. This is precisely the bounded-noise condition required by the self-normalised least-squares bounds of Abbasi-Yadkori et al. [1].

Collecting these observations over the sample set C yields, for each expression $f \in \Lambda$, a family of Bernoulli trials with common mean $f[\mathbf{u}] = c_f + b_f^\top \mathbf{u}$. Since all transition triples in $\mathrm{Occ}(f)$ are governed by the same symbolic expression, they provide information about the same unknown linear quantity and can therefore be pooled. This gives a single count-based regression equation per distinct expression rather than per concrete transition triple, which is exactly the least-squares analogue of the parameter-tying mechanism used in our interval-based constructions. Aggregating these pooled equations over all $f \in \Lambda$ then yields the following ridge least-squares system.

$$V = \lambda I + \sum_{f \in \Lambda} N_f \, b_f b_f^\top, \qquad r = \sum_{f \in \Lambda} b_f \big(K_f - N_f c_f\big),$$

with estimator

$$\hat{\mathbf{u}} = V^{-1} r.$$

Here, V is the regularised design matrix and $\lambda > 0$ is the ridge parameter. The self-normalised concentration result of Abbasi-Yadkori et al. [1], instantiated in the present count-based setting, yields the following sound confidence ellipsoid

$$\mathcal{E}_\delta := \left\{ \mathbf{v} \in \mathbb{R}^\ell \mid (\mathbf{v} - \hat{\mathbf{u}})^\top V (\mathbf{v} - \hat{\mathbf{u}}) \le \beta^2 \right\},$$

where

$$\beta = R\sqrt{\log \frac{\det(V)}{\lambda^\ell} + 2 \log \frac{1}{\delta}} + \sqrt{\lambda}\, S,$$

where $1 - \delta \in (0, 1)$ is the overall confidence. With this construction, the unknown instantiation $\mathbf{u}$ lies in $\mathcal{E}_\delta$ with probability at least $1 - \delta$ over the random sample set C [4]. The constant R is the sub-Gaussian noise bound, which for Bernoulli transition observations is $R = \frac{1}{2}$. The constant S is any a priori bound on $\|\mathbf{u}\|_2$, for $D \subseteq [0, 1]^\ell$, the choice $S = \sqrt{\ell}$ is always sound, and $\lambda > 0$ is an arbitrary ridge parameter, which we choose as $\lambda = 10^{-2}$ to stabilise the regression.

The ellipsoidal baseline induces, for each (s, a), the uncertainty set

$$\mathcal{P}_{R(\mathcal{E})}(s, a) := \{\mu \in \Delta(S) \mid \exists \mathbf{v} \in \mathcal{E}_\delta : \mu(s') = P[\mathbf{v}](s, a, s') \text{ for all } s' \in S\}.$$

The inner optimisation in the robust Bellman update in Eq. (2) then becomes:

$$\min_{\mathbf{v}\in\mathcal{E}_\delta} \sum_{s'\in S} P[\mathbf{v}](s,a,s')\, V_U^*(s').$$

Thus, robust synthesis under the ellipsoidal baseline requires solving a convex conic optimisation problem for each Bellman backup.

As in the polytopic case, the relaxations of Section 3.1 can also be applied to ellipsoidal uncertainty sets in order to obtain interval MDPs and thereby enable efficient robust synthesis. The difference lies in the optimisation problems used to construct these relaxations: instead of solving linear programmes for each expression in the expression-wise projection, or for each parameter in the parameter-wise projection, we solve second-order cone programmes over the ellipsoidal uncertainty region. The resulting interval model is again a sound rectangular over-approximation of the original ellipsoidal uncertainty set, and the inclusion hierarchy of Theorem 2 carries over accordingly.

Overall, this ellipsoidal construction is a natural adaption of the self-normalised least-squares confidence sets [4] to the pMDP setting. In the extended experimental evaluation in Appendix F, we compare this ellipsoidal baseline with the polytopic confidence regions and their corresponding relaxations across the considered case studies.

F Extended Experimental Evaluation

In this appendix, we provide detailed descriptions of the considered case studies, as well as extended results complementing those in Section 4. In particular, we report a refined breakdown of runtime into time spent constructing and solving the respective uncertain models, and include additional results for further uncertainty classes, namely ellipsoids, as an additional baseline for comparison [4, 49].

F.1 Descriptions of Benchmark Environments

Aircraft collision avoidance. This benchmark models two aircraft moving towards each other on a discrete grid of size $N \times M$, where our agent controls one aircraft and aims to reach the far end of the grid without entering a collision zone around the adversarial aircraft [26]. In each step, the agent chooses to ascend, descend, or maintain altitude while moving forward, and the adversarial aircraft simultaneously moves forward and may also change altitude. The transition probabilities are governed by four unknown mixture parameters $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4\}$, which combine four position-dependent motion modes. Concretely, the parameters determine both the success probability of the agent's vertical manoeuvre and the distribution of the adversary's vertical movement, with all probabilities varying affinely with the horizontal position along the grid. The resulting model therefore has a linear parametric structure with four dependent mixture weights. Our

objective is to maximise the probability of reaching the goal without collision. We consider instances of sizes $(N, M) \in \{(50, 10), (100, 20), (200, 40)\}$.

Betting game. This benchmark models a sequential betting process in which the agent starts with 10 coins and, over a horizon of n rounds, repeatedly chooses one of five actions corresponding to betting 0, 1, 2, 5, or 10 coins [8]. After each non-zero bet, the agent either wins the wager and gains the bet amount, or loses it and forfeits the same amount. The transition probabilities are governed by two unknown parameters: a base win probability θ and a state-dependent term proportional to the current amount of money. In particular, for larger bets, the win probability is an affine function of both θ and the current capital, which yields a linear parametric structure with two parameters. The objective is to maximise the expected amount of money after n betting rounds. We consider instances with horizons $n \in \{50, 100, 150\}$.

Engagement. This benchmark models a customer-engagement process on a ladder of levels $\{0, \ldots, L\}$, where level 0 represents churn and level L a successful conversion or purchase. At each decision step, the agent chooses one of three interventions, i.e., *light*, *medium*, or *aggressive*, which may increase, decrease, or leave unchanged the current engagement level. The transition probabilities are governed by five unknown mixture parameters $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$, with $\theta_5 = 1 - (\theta_1 + \theta_2 + \theta_3 + \theta_4)$, which combine five latent customer-response types. The effect of an intervention further depends on the current engagement zone (cold, warm, or hot), as well as on the previously chosen action and a short cooldown memory that penalises repeated aggressive interventions. The resulting model therefore has a linear parametric structure with five dependent mixture weights and non-trivial state-dependent dynamics. Rewards combine a per-step time cost, action-dependent intervention costs, and a penalty for churn. The objective is to minimise the expected accumulated cost until either churn or purchase is reached. We consider instances with $L \in \{100, 300, 1000\}$.

Mars rover. This benchmark is a richer variant of the semi-autonomous vehicle domain of [25], and is already introduced in Section 1. A rover moves on an $X \times Y$ grid and aims to reach a target location while maintaining communication with a remote controller via two unreliable channels. At each step, it may either move in one of the allowed directions or attempt communication over one of the two channels, with task failure occurring if too many moves are made without a successful transmission. The transition probabilities are governed by six unknown mixture parameters $\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$, which combine five position-dependent channel modes. In contrast to the original benchmark, our variant includes a larger grid, a richer spatial structure with zone-dependent scaling and local jamming regions, and a larger communication budget, yielding a more varied but still linear parametric structure. We consider instances of sizes $(X, Y) \in \{(50, 50), (75, 75), (125, 125)\}$.

Glider. This benchmark models an autonomous underwater glider navigating on a two-dimensional grid from a start location to a goal region in the presence

of ocean currents [15]. At each location, the horizontal and vertical current components are known, but their effect on the glider's movement is governed by two unknown parameters, θ_h and θ_v, capturing the sensitivity to horizontal and vertical drift, respectively. When the glider chooses a movement direction, the commanded motion may fail along the intended axis and may additionally drift along the orthogonal axis, yielding up to four possible outcomes per action. Because these two effects combine multiplicatively, the transition probabilities contain bilinear terms in θ_h and θ_v, giving rise to a non-linear parametric structure. The objective is to minimise the expected time to reach the goal. We consider instances of sizes $(X, Y) \in \{(21, 17), (51, 47), (106, 99)\}$.

Parallel betting game. This benchmark is a parallel composition of two betting games that are played simultaneously over a common horizon of n rounds. In each round, the agent chooses a single betting action, corresponding to wagering 0, 1, 2, 5, or 10 coins, and this action is applied independently to both games. Each game maintains its own capital and is governed by separate unknown parameters, determining its base win probability. As in the single-game benchmark, larger bets have outcome probabilities that additionally depend on the current amount of money. Since the overall state combines the capitals of both games and the final reward is their sum, the resulting model captures a coupled decision problem with two unknown parameters. Moreover, because the transition probabilities in the second game depend multiplicatively on θ_2 and the current capital, the parallel composition yields a non-linear parametric structure. The objective is to maximise the expected total amount of money after n rounds. We consider instances with horizons $n \in \{5, 10, 20\}$.

F.2 Extended Results for Uniform Sampling

Table 3 reports the extended results for the first setting, in which 10^5 trajectories are sampled uniformly and used to compare the tightness of the resulting uncertain models. In addition to the overall results, we separate the total runtime into the time spent constructing the uncertain model and the time spent solving it.

Table 3: Comparison of obtained bounds and runtimes for both building and solving the respective models obtained after sampling 10^5 trajectories uniformly.

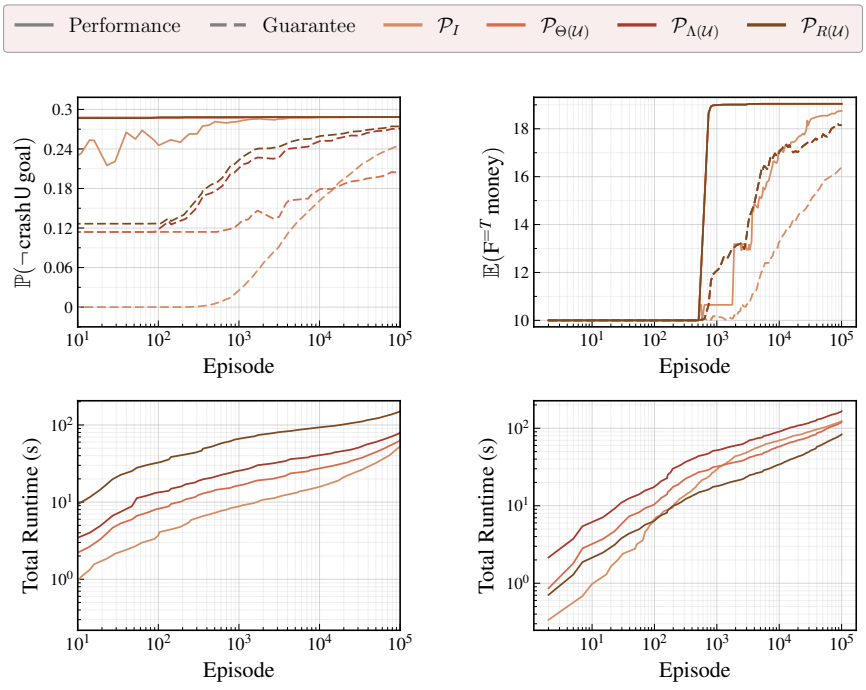
Benchmark	Instance	$\mathcal{P}_I$				$\mathcal{P}_{\Theta(\mathcal{U})}$				$\mathcal{P}_{\Lambda(\mathcal{U})}$				$\mathcal{P}_{R(\mathcal{U})}$			
		$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**
Aircraft	(50, 10)	[0.67, 0.814]	0.19	0.77	3.28	[0.618, 0.828]	0.28	1.29	3.34	[0.719, 0.767]	0.06	2.17	3.55	[0.725, 0.761]	**0.05**	44.95	2.17
	(100, 20)	[0.745, 0.906]	0.19	4.11	26.34	[0.679, 0.918]	0.29	8.77	25.94	[0.805, 0.865]	0.07	9.26	30.28	[0.814, 0.858]	**0.05**	309.67	43.69
	(200, 40)	[0.628, 0.92]	0.36	43.23	645.05	[0.55, 0.928]	0.47	114.09	763.95	[0.754, 0.85]	0.12	128.50	754.28	[0.769, 0.838]	**0.09**	2175.51	2288.73
Betting Game	(50)	[14.4, 65.3]	1.88	0.92	1.79	[25.8, 28.7]	**0.10**	1.57	4.92	[25.8, 28.7]	**0.10**	4.67	2.87	[25.8, 28.7]	**0.10**	1.28	1.01
	(100)	[16.5, 162]	3.24	3.10	9.98	[41.8, 47.7]	**0.13**	4.99	7.87	[41.8, 47.7]	**0.13**	16.87	9.44	[41.8, 47.7]	**0.13**	8.79	8.26
	(150)	[17.5, 263]	3.98	5.69	19.49	[56.5, 65.5]	**0.15**	19.36	18.61	[56.5, 65.5]	**0.15**	35.29	17.87	[56.5, 65.5]	**0.15**	13.10	25.24
Engagement	(100)	[38.4, 49.3]	0.26	0.12	5.33	[16.6, 5422]	127.35	0.11	4.33	[39.8, 45]	**0.12**	0.19	0.82	[39.8, 45]	**0.12**	1.24	0.58
	(300)	[38.6, 46.8]	0.19	0.32	16.23	[18.7, 3615]	84.74	0.31	20.98	[40.8, 45.3]	**0.11**	0.48	7.47	[40.8, 45.3]	**0.11**	1.41	5.86
	(1000)	[39.1, 45.4]	0.15	0.98	69.04	[24, 687]	15.63	0.91	56.83	[41.3, 43.8]	**0.06**	1.26	64.54	[41.3, 43.8]	**0.06**	2.17	93.27
Mars Rover	(50, 50)	[0, 0.797]	1.59	9.38	82.64	[0.177, 0.833]	1.31	21.09	115.78	[0.488, 0.515]	**0.05**	27.74	87.41	–	–	TO	TO
	(75, 75)	[0, 0.895]	1.42	15.96	505.71	[0.0904, 0.9]	1.28	47.80	529.12	[0.619, 0.647]	**0.04**	61.38	518.83	–	–	TO	TO
	(125, 125)	[0, 0.877]	1.42	19.64	2149.57	[0.0441, 0.922]	1.42	97.87	2455.97	[0.599, 0.635]	**0.06**	111.17	2195.12	–	–	TO	TO
Glider	(21, 17)	[41.1, 44.6]	0.08	0.16	0.09	[42.4, 42.9]	0.01	0.16	0.03	[42.5, 42.8]	0.01	0.65	0.06	[42.5, 42.8]	**0.01**	37.20	0.06
	(51, 47)	[62.3, 20001]	293.07	0.52	28.40	[58.8, 59.2]	0.01	1.12	0.55	[58.8, 59.1]	0.00	16.59	0.57	[67.9, 68.1]	**0.00**	9266.65	1.58
	(106, 99)	[66.4, 35254]	464.04	0.52	52.43	[75.6, 76.1]	0.01	1.91	1.97	[75.6, 76]	**0.00**	46.57	1.94	–	–	TO	TO
Parallel Betting	(5)	[24.3, 44.7]	0.64	1.89	0.69	[25.1, 28.2]	0.10	0.38	0.02	[26.1, 27.2]	0.04	0.77	0.03	[26.1, 27.2]	**0.03**	10.13	0.03
	(10)	[26.4, 40.6]	0.44	2.13	0.73	[29.7, 34.5]	0.15	2.83	0.40	[31.1, 32.9]	0.06	5.79	0.63	[31.1, 32.8]	**0.05**	116.11	0.41
	(20)	[13, 110]	3.39	16.00	5.24	[25.4, 31.7]	**0.22**	659.37	6.94	–	–	TO	TO	–	–	TO	TO

Table 4: Salient characteristics of the benchmarks for the online learning setting.

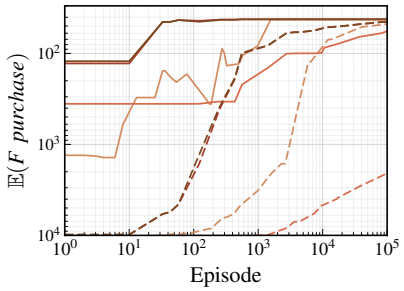
Benchmark	Instance	$\lvert S\rvert$	$\lvert T\rvert$	$\lvert\Theta\rvert$	$\lvert\Lambda\rvert$	Prop.
Aircraft	(20, 10)	1833	23743	6	271	$\mathbb{P}$
Betting Game	(25, 0.55)	3625	27703	2	1283	$\mathbb{E}$
Engagement	(50, 0.30)	994	2962	5	61	$\mathbb{E}$
Mars Rover	(10, 10)	5139	24515	6	145	$\mathbb{P}$
Glider	(11, 9)	99	1317	4	225	$\mathbb{E}$
Parallel Betting	(6, 0.55)	5046	30746	3	597	$\mathbb{E}$

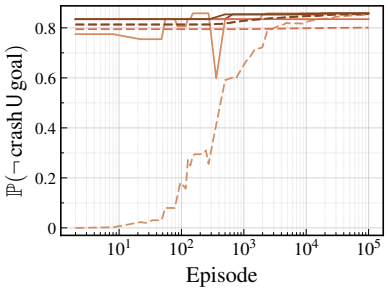
F.3 Extended Learning Results

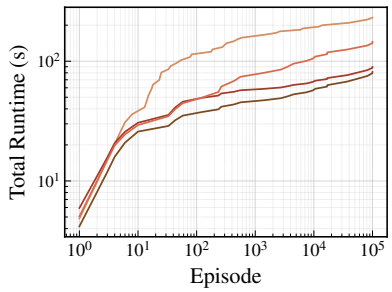
This section presents the full results for the online learning experiments. Table 4 summarises the salient characteristics of the benchmark instances used in this setting. Since the uncertain models must be rebuilt and resolved repeatedly in order to update the sampling policy, these instances are smaller than those used in the uniform-sampling setting, where each model is solved only once. Overall, the results show that exploiting the parametric structure consistently yields more effective policies and stronger certified performance guarantees from the same data and under the same high-confidence PAC guarantees.

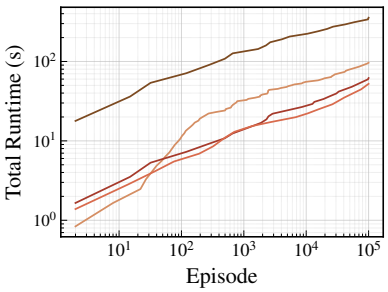


(c) Aircraft (d) Betting Game

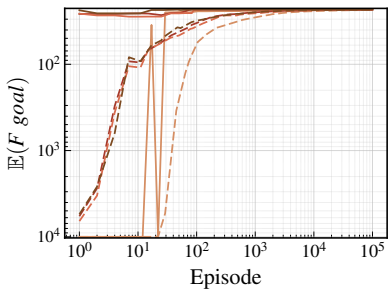


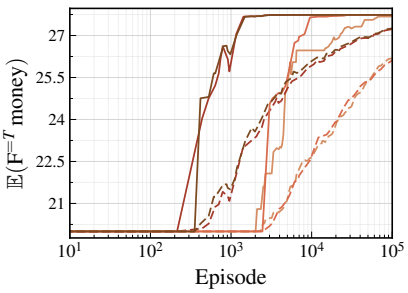


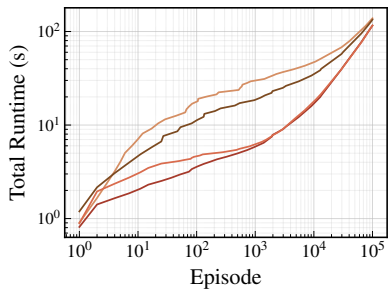


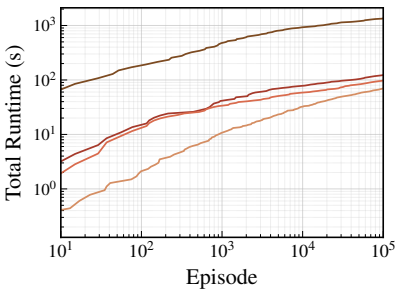


(e) Engagement (f) Mars Rover









(g) Glider (h) Parallel Betting

Fig. 4: Extended results for robust policy learning. For each benchmark, the upper plots compare performance and guarantees, and the lower plots show the total runtimes of each method.

F.4 Experimental Results with Ellipsoidal Uncertainty Sets

This section reports the results obtained when instantiating our learning and solving framework with the ellipsoidal uncertainty sets discussed in Appendix E. Table 5 summarises the results for the uniform-sampling setting, comparing runtimes and the resulting certified guarantees across the different uncertainty sets and relaxations. We omit the local rectangular relaxation $\mathcal{P}_{R(\mathcal{E})}$ from the table, as it timed out on all considered case studies. This is due to the high computational cost of solving a second-order cone programme for every state–action pair in every iteration of robust value iteration. This highlights the importance of the interval-based relaxations, which make ellipsoidal uncertainty practically usable by projecting it back to tractable interval uncertainty sets.

Overall, the results indicate that the approach based on projecting confidence intervals into the parameter space consistently yields tighter uncertainty sets and stronger guarantees than the ellipsoidal baselines.

Table 5: Comparison of obtained bounds and runtimes for both building and solving the respective models obtained after sampling 10^5 trajectories uniformly. This includes both polytopic and ellipsoidal uncertainty sets and their relaxations.

Benchmark	**Instance**	$\mathcal{P}_I$				$\mathcal{P}_{\Theta(\mathcal{U})}$				$\mathcal{P}_{\Theta(\mathcal{E})}$				$\mathcal{P}_{\Lambda(\mathcal{U})}$				$\mathcal{P}_{\Lambda(\mathcal{E})}$				$\mathcal{P}_{R(\mathcal{U})}$			
		$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**	$[\underline{\mathbf{V}}, \overline{\mathbf{V}}]$	**Rel. gap**	**Build [s]**	**Solve [s]**
Aircraft	(50, 10)	[0.67, 0.814]	0.19	0.30	0.43	[0.618, 0.828]	0.28	0.95	0.60	[0.599, 0.835]	0.32	0.50	0.50	[0.719, 0.767]	0.06	1.13	0.60	[0.71, 0.78]	0.09	1.86	0.63	[0.725, 0.761]	**0.05**	28.02	0.54
	(100, 20)	[0.745, 0.906]	0.19	2.02	16.54	[0.679, 0.918]	0.29	6.56	27.81	[0.678, 0.919]	0.29	3.14	14.67	[0.805, 0.865]	0.07	7.47	19.30	[0.805, 0.866]	0.07	28.12	28.47	[0.814, 0.858]	**0.05**	217.95	20.55
	(200, 40)	[0.628, 0.92]	0.36	15.71	282.38	[0.55, 0.928]	0.47	54.06	327.19	[0.575, 0.926]	0.44	22.96	266.26	[0.754, 0.85]	0.12	61.25	253.13	[0.768, 0.844]	0.09	1667.3	344.75	[0.769, 0.838]	**0.09**	1438.33	754.25
Betting Game	(50, 0.55)	[12.1, 83.7]	2.65	0.42	0.76	[25.2, 28.5]	**0.12**	0.90	1.17	[24.6, 29.5]	0.18	0.58	1.42	[25.2, 28.5]	**0.12**	1.80	1.04	[24.6, 29.5]	0.18	25.42	1.30	[25.2, 28.5]	**0.12**	0.91	0.50
	(100, 0.55)	[16.5, 162]	3.24	1.48	5.36	[41.8, 47.7]	**0.13**	4.36	6.70	[40.9, 49.5]	0.19	2.53	8.04	[41.8, 47.7]	**0.13**	7.96	11.16	[40.9, 49.5]	0.19	584.54	6.64	[41.8, 47.7]	**0.13**	4.43	4.74
	(150, 0.55)	[17.5, 250]	3.91	2.40	12.99	[54.7, 63.3]	**0.15**	9.34	9.67	[53.6, 66.1]	0.21	5.26	15.28	[54.7, 63.3]	**0.15**	15.75	14.86	[53.6, 66.1]	0.21	1916.81	5.58	[54.7, 63.3]	**0.15**	9.16	8.65
Engagement	(100, 0.30)	[38.4, 49.3]	0.26	0.07	2.40	[16.6, 5422]	127.35	0.10	1.58	[15.3, 9740]	229.13	0.10	2.05	[39.8, 45]	**0.12**	0.13	0.46	[35.6, 53.2]	0.41	0.13	0.49	[39.8, 45]	**0.12**	0.74	0.31
	(300, 0.30)	[38.6, 46.8]	0.19	0.16	5.92	[18.7, 3615]	84.74	0.24	6.57	[15.7, 9110]	214.27	0.20	6.90	[40.8, 45.3]	**0.11**	0.28	3.34	[36.7, 50.6]	0.33	0.26	3.04	[40.8, 45.3]	**0.11**	0.86	2.28
	(1000, 0.30)	[39.1, 45.4]	0.15	0.50	20.16	[24, 687]	15.63	0.67	20.46	[16, 7857]	184.74	0.45	20.89	[41.3, 43.8]	**0.06**	0.69	20.60	[38.5, 47.7]	0.22	0.56	19.95	[41.3, 43.8]	**0.06**	1.28	40.77
Mars Rover	(50, 50)	[0, 0.797]	1.59	3.22	46.72	[0.177, 0.833]	1.31	6.26	44.63	[0.174, 0.837]	1.32	5.03	46.19	[0.488, 0.515]	**0.05**	7.07	42.73	[0.476, 0.525]	0.10	23.26	43.82	–	–	TO	TO
	(75, 75)	[0, 0.895]	1.42	6.35	240.78	[0.0904, 0.9]	1.28	19.94	232.96	[0.0801, 0.901]	1.30	15.01	216.61	[0.619, 0.647]	**0.04**	21.99	235.26	[0.605, 0.658]	0.08	108.24	217.49	–	–	TO	TO
	(125, 125)	[0, 0.877]	1.42	15.65	1279.81	[0.0441, 0.922]	1.42	114.78	1326.19	[0.0422, 0.922]	1.43	57.16	1590.10	[0.599, 0.635]	**0.06**	116.29	1384.91	–	–	TO	TO	–	–	TO	TO
Glider	(21, 17)	[41.1, 44.6]	0.08	0.07	0.04	[42.4, 42.9]	**0.01**	0.15	0.05	[42, 43.3]	0.03	0.23	0.04	[42.5, 42.8]	**0.01**	0.30	0.04	[42.2, 43.1]	0.02	2.15	0.04	[42.5, 42.8]	**0.01**	21.90	0.03
	(51, 47)	[89.8, 22443]	221.40	0.17	11.20	[100.66, 101.21]	0.01	0.82	0.27	[100, 102]	0.01	0.34	0.28	[100.76, 101.12]	**0.00**	5.97	0.27	[100.57, 101.32]	0.01	11.47	0.27	[100.76, 101.12]	**0.00**	5782.72	0.15
	(71, 67)	[87, 30071]	288.64	0.11	20.00	[103, 105]	**0.01**	1.82	0.60	[102, 105]	0.03	0.66	0.65	[103, 104]	**0.01**	21.50	0.61	[103, 105]	0.02	49.26	0.50	–	–	TO	TO
Parallel Betting	(5, 0.55)	[24, 29.4]	0.20	0.14	0.03	[25.1, 28.2]	0.12	0.41	0.03	[23.1, 30.6]	0.28	0.35	0.03	[26.1, 27.2]	**0.04**	0.49	0.03	[24.9, 28.6]	0.14	1.71	0.03	[26.1, 27.2]	**0.04**	6.84	0.03
	(10, 0.55)	[26.4, 40.6]	0.44	1.06	0.44	[29.7, 34.5]	0.15	2.59	0.44	[26.9, 37.9]	0.34	1.79	0.45	[31.1, 32.9]	0.06	3.12	0.44	[29.3, 34.9]	0.17	25.91	0.21	[31.1, 32.8]	**0.05**	73.99	0.21
	(20, 0.55)	[13, 110]	3.39	5.56	4.81	[25.4, 31.7]	**0.22**	898.14	4.48	[22.6, 35.8]	0.46	21.96	3.22	–	–	TO	TO	–	–	TO	TO	–	–	TO	TO