Polynomial-Time Verification of PCTL Properties of MDPs with Convex Uncertainties

Alberto Alessandro Angelo Puggelli

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-118
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-118.html

May 20, 2014

Polynomial-Time Verification of PCTL Properties of MDPs with Convex Uncertainties

by Alberto Alessandro Angelo Puggelli

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Koushik Sen
Research Advisor

(Date)

* * * * * * *

Professor Alberto L. Sangiovanni-Vincentelli
Second Reader

(Date)

To me, Beauty is the wonder of wonders.
It is only shallow people
who do not judge by appearances.

Oscar Wilde, *The Picture of Dorian Gray*

Abstract

We address the problem of verifying Probabilistic Computation Tree Logic (PCTL) properties of Markov Decision Processes (MDPs) whose state transition probabilities are only known to lie within uncertainty sets.

We first introduce the model of Convex-MDPs (CMDPs), i.e., MDPs with convex uncertainty sets. CMDPs generalize Interval-MDPs (IMDPs) by allowing also more expressive (convex) descriptions of uncertainty.

Using results on strong duality for convex programs, we then present a PCTL verification algorithm for CMDPs, and prove that it runs in time polynomial in the size of a CMDP. This result allows us to lower the previously known algorithmic complexity upper bound for IMDPs from co-NP to PTIME, and it is valid also for more expressive (convex) uncertainty models.

We validate the proposed approach on two case studies: the verification of a consensus protocol and of a dynamic configuration protocol for IPv4 addresses.

A concise version of the results presented in this thesis can be found in [1]:

A. Puggelli, W. Li, A. L. Sangiovanni-Vincentelli, S. A. Seshia, *Polynomial-Time Verification of PCTL Properties of MDPs with Convex Uncertainties*, Computer Aided Verification, 2013.

Acknowledgments

I consider the journey that led to the results presented in this thesis as unusual, surprising and, to some extent, exciting. Or at least this is how I lived it. Starting from a class project, I dove into a body of material that was completely new to me and that I had to discover and understand mainly by myself. My stubbornness made me continue the research on this field also after the class was over, and my naiveness on the material (and a great deal of good luck) led me to take a different approach to tackle a problem that had been investigated for the previous ten years, and devise the first-known polynomial time algorithm to solve it.

I would first like to thank my mates along this journey, Wenchao Li, John B. Finn and prof. Seshia. Our "productive" meetings saw the development of a big part of the skeleton of the results presented in this thesis, and our conversations have been the closest to what I consider the ideal "maieutic" approach. The knowledge was inside us, and we discovered it through a sequence of well-posed questions.

I would then like to thank my two advisors, prof. Sangiovanni-Vincentelli and prof. Alon, for supporting me and this activity, although it was taking time away from my main research projects. Their love for research crossed the boundaries of the specific projects I was assigned to work on, and the results have been proven to be successful for everybody.

I would also like to thank prof. El Ghaoui, for introducing me to the fundamentals of convex optimization, and for giving the following precious piece of advise: "If you are faced with a convex problem, take its dual and things will get simpler". This could be a nice and witty summary of how the presented results were discovered. Thank you also to prof. Sen. It was a pleasure for me to have one of the first researchers in the field be my research advisor for this thesis.

I will soon write a longer "acknowledgment" section on my Ph.D. dissertation in which I will have time to thank everybody that I have had the pleasure to work with in these last few years, which the work presented in this thesis only represents a small fraction of. Nevertheless, I would like to thank Eleonora also here, for being next to me every day while developing this work and for understanding every time I had to spend my weekends working instead of doing things together. Her help cannot be matched and a large part of this success is to be shared with her.

Finally, I would like to thank my parents and my sister for the unconditional support during the months in which I was working on the project. The time spent on it gave me even less time to share with them, but they have always managed to make me feel close to them, despite the geographical distance. And a special thank you to my Mother for coming to St. Petersburg after I presented this work at CAV 2013 to spend some time with me, given the difficulties of staying together during the year. We had a great time together!

Contents

List of Figures

List of Tables

Chapter 1
Introduction

In this chapter, we present the motivations that drove the work described in the thesis. We then summarize the main achieved contributions and outline the content of the rest of the thesis.

1.1 Motivations

In the last decade, industrial and commercial electronics has been driven towards two only-at-first-glance opposite directions. The infinitesimally small, with single-digit nanometric devices, enabling always-increasing computational resources at low power consumption. And the immensely huge, with networks of billions of devices connecting the whole world, enabled by the larger availability of miniaturized and energy efficient devices. Indeed, embedded electronic systems are ubiquitous, both in the environment around us and in the gadgets (often smart-phones) that we always bring along with us. This situation is foreseen to further expand, and a swarm of sensors connected to the IT cloud is predicted to surround us from the early 2020s, when it is estimated that there will be more then a 1000 electronic devices per person. While these advances will enable a massive increase of the availability of information around us, a key challenge to be faced to make this extraordinary scenario true will be managing the increased complexity in the design, verification and operation of these systems.

This thesis will focus on formal techniques for the verification of the behavior of stochastic systems. Formal verification is a field in rapid expansion because it addresses the need of exhaustively validating the correct functionality of systems that exhibit too many behaviors (due to their complexity) to be fully tested through simulation. Since the cost of putting in the market a faulty product, and then recall it to fix it, is unbearable for most businesses, formal verification is playing an increasingly bigger role in the development flow of electronic systems. While formal verification for deterministic systems have already reached a reasonable level of maturity (although some level of domain-specific expertise is still often required to abstract the main system behaviors and improve the scalability of the analysis), the focus of the research community is now shifting towards stochastic systems.

System behavior can be considered stochastic either because of the presence of actual randomization (for example, random back-off schemes in wireless transmission protocols), or as a modeling tool to abstract complex deterministic behaviors (for example, the electricity power demand of a city neighborhood can be captured by its mean and standard deviation varying during the day, since capturing the actual deterministic data would be too computationally expensive). Given the importance of such systems, stochastic verification techniques are required.

Stochastic models like Discrete-Time Markov Chains (DTMCs) [2] and Markov Decision Processes (MDPs) [3] have been used to formally represent systems that exhibit random or probabilistic behaviors. These systems need *quantitative* analysis [4] to answer questions such as “what is the probability that a request will be eventually served?”. Properties of these systems can be expressed and analyzed using logics such as Probabilistic Computation Tree Logic (PCTL) [5] — a probabilistic logic derived from CTL which includes a probabilistic operator P — as well as techniques for probabilistic model checking [6].

One critical step in the modeling of probabilistic systems is the estimation of state transition probabilities. When modeling randomized protocols (e.g., a randomized consensus protocol), transition probabilities are usually inferred from the ideal behavior of the system. For example, a probability of 0.5 is assigned to the two possible results of a coin toss. On the other hand, when modeling a stochastic physical system (e.g., the quality of a wireless channel), transition probabilities are inferred by performing a measurement campaign and by computing the occurrence frequencies of each possible (discretized) observation of the physical phenomenon. For example, if 4 out of 10 sent packages across a wireless channel drop, the probability of successful transmission will be 0.6. While widely adopted for their intrinsic simplicity and computational tractability, these techniques for the estimation of transition probabilities might fail to capture some critical behavior of the system under analysis. For randomized protocols, faulty agents or agents under a security attack might fail to behave as in the ideal scenario. Continuing the example of the coin toss, this would translate into an equivalent “biased” coin toss, in which, for example, expected probabilities might be 0.4 for “tail” and 0.6 for “head”. In the estimation of physical processes, the finite precision in taking measurements and the finite number of measurements that can be taken in a practical scenario limit the accuracy of the inferred transition probabilities.

In fact, formal statistical techniques to capture this uncertainty in the estimation of transition probabilities do exist and they have been widely studied in the statistics and optimal control community [7]. Most of these techniques assume that the transition probabilities are not known with precision but only lie in an uncertainty set of potentially observable probabilities. To keep computation tractable, these uncertainty sets are usually convex, e.g., a closed interval, an ellipsoid or a likelihood region. Also in the verification community, the concept of transition uncertainty has been proposed. Interval-valued Discrete-Time Markov Chains (IDTMCs) have been introduced to capture modeling uncertainties [8]. IDTMCs are DTMC models where each transition probability is assumed to lie within a compact range. Two semantic interpretations have been proposed for these models [9]: Uncertain Markov Chains (UMCs) and Interval Markov Decision Processes (IMDPs). An UMC is interpreted as a family of (possibly uncountably many) DTMCs, where each member of the family is a DTMC whose transition probabilities lie within the interval range given in the UMC. In IMDPs, the uncertainty is resolved through non-determinism. Each time a state is visited, a transition distribution within the interval constraints is adversarially picked, and a probabilistic step is taken accordingly. Thus, IMDPs allow modeling a non-deterministic choice made from a set of (possibly uncountably many) choices. For both semantics, the verification problem amounts to determine whether a desired property holds also under the worst-case resolution of uncertainty. In this thesis, we do not consider UMCs and focus on IMDPs. From a modeling standpoint, IMDP semantics represents the worst-case scenario of UMCs, since in IMDPs a new adversarial state transition probability distribution is chosen at each step, while in UMCs the adversarial transition probability distribution is chosen only once. Further the development of verification algorithms for UMCs has been proven in [10] to be harder than for IMDPs, so UMCs are less amenable to a scalable analysis.

An upper-bound on the complexity of model checking PCTL properties on IMDPs was previously shown to be PSPACE [9], and the result was later improved to co-NP [10]. These results rely on the construction of an equivalent MDP that encodes all behaviors of the IMDP. For each state in the new MDP, the set of transition probabilities is equal to the Basic Feasible Solutions (BFS) of the set of inequalities specifying the transition probabilities of the IMDP. Since in the worst case the number of BFS is exponential in the number of states in the IMDP, the equivalent MDP can have size exponential in the size of the IMDP. Given the aforementioned results, no computationally efficient algorithm was known for the verification of formal properties of systems whose transition probabilities were captured in uncertainty sets.

An interval model of uncertainty may appear to be the most intuitive to analyze. However, there may be significant advantages in being able to accommodate more expressive (and less pessimistic) uncertainty sets in addition to intervals. In [11], a financial portfolio optimization case-study is analyzed in which uncertainty arises from estimating the rate of return for each asset. The authors claim that the interval model of uncertainty is too conservative in this scenario, because it would suggest to invest the whole capital into the asset with the largest worst-case return. The ellipsoidal model of uncertainty proposed in that paper returns instead a solution that spreads the capital across multiple assets, a more profitable strategy in the long run. As a further example, the authors of [12] consider the problem of estimating transition probabilities from measurements. If the probabilities $\boldsymbol{x}$ are normally distributed with mean $\boldsymbol{\mu}$, and covariance matrix Σ, the ellipsoid defined by $(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \leq 1$ is a valuable approximation that eliminates "almost" impossible outliers, which would otherwise be included by an interval approximation. Further, depending on the field, researchers use different models to represent uncertainty. Maximum likelihood models are often used, for example, to estimate chemical reaction parameters [13]. Entropy models are instead used when the available data points are limited or it is costly to collect them, e.g. in text segmentation [14] and political vote prediction [15].

1.2 Main Contributions

In this thesis, we present three main contributions.

First, we present the first-known *polynomial-time algorithm* (in both size of the model and size of the formula) for the verification of the same fragment of PCTL considered in [9,10] (the *Bounded Until* operator is disallowed). This shows that the problem is in the complexity class PTIME (also known as P). We then extend the algorithm to full PCTL (with *Bounded Until*), and show that its time complexity only increases to pseudo-polynomial in the maximum integer time bound in *Bounded Until*.

Second, in order to increase the expressiveness of the model under verification, we introduce the model of *Convex-MDP (CMDP)*, i.e., an MDP whose state transition probabilities are only known to lie within convex uncertainty sets. We show that the algorithms proposed in this paper can be extended to verify CMDP for a wide and relevant class of convex uncertainty sets (e.g., sets expressed with the ellipsoidal and likelihood models of uncertainty), while maintaining the same complexity results proven for IMDPs. Moreover, different models of uncertainty can be used within the same CMDP to represent different sources of uncertainty, thus further increasing the expressiveness of the proposed approach. We also note that the complexity results presented in [9] and [10] cannot be trivially extended to verifying CMDPs. This is because BFS are not defined for generic convex inequalities, so the construction of an equivalent MDP would not be possible. The complexity results are compared in Table 1.

Finally, we demonstrate the relevance of our approach with two case studies. First, we study the behavior of a distributed stochastic consensus protocol when one of

Table 1: Known Upper-Bound on the Complexity of PCTL Model Checking.

Model	DTMC [5]	IMDP [10]	IMDP/CMDP [ours]
Complexity	PTIME	co-NP	PTIME

the processes participating in the protocol is faulty or behaving maliciously. Second, we analyze the performance of a wireless protocol for domotic applications operating across a lossy link. Results show that a small uncertainty in the transition probabilities indeed yields a significant change in the verification results, thus revealing the importance of the proposed analysis. Further, we demonstrate the runtime scalability of the proposed approach to the analysis of problems of practical relevance.

1.3 Outline

The rest of the thesis is organized as follows.

In **Chapter 2**, we introduce the framework used in this work to model the behavior of probabilistic systems. We first formally introduce the concept of Convex-MDPs. We then describe the convex models of uncertainty analyzed in this work. Finally, we define the Probabilistic Computation Tree Logic, i.e., the logic used to express properties of Convex-MDPs.

In **Chapter 3**, we survey related work in the literature both in the fields of formal verification and robust control.

In **Chapter 4**, we give an overview of the proposed approach to verify PCTL properties of CMDPs, and formalize the new theoretical complexity results.

In **Chapter 5**, we give details of the proposed verification algorithm and prove that it runs with algorithmic complexity polynomial in the size of the model and of the specification.

In **Chapter 6**, we describe two case studies analyzed with the proposed approach — a randomized consensus protocol and a dynamic configuration protocol for domotic applications.

Lastly, we conclude and discuss future research directions in **Chapter 7**.

Chapter 2

A Framework to Model Probabilistic Systems

In this chapter, we introduce the framework used in this work to model the behavior of stochastic systems. We begin with preliminary definitions of concepts that will be used in the subsequent derivations. We then introduce the model of Convex-MDPs, i.e., Markov Decision Processes whose state transitions are only known to lie in convex uncertainty sets, and provide details of the mathematical representation of four convex uncertainty sets commonly used in statistics. Finally, we conclude the chapter by introducing Probabilistic Computation Tree Logic, i.e., the formal logic that we use to express properties of Convex-MDPs.

We give the definition of Probability Distribution (PD), of convex set and of convex function.

Definition 2.1. *A PD over a finite set Z of cardinality n is a vector $\mu \in \mathbb{R}^n$ satisfying $\mathbf{0} \leq \mu \leq \mathbf{1}$ and $\mathbf{1}^T\mu = 1$. The element $\mu[i]$ represents the probability of realization of the event z_i. We call $Dist(Z)$ the set of distributions over Z.*

Definition 2.2. *A set C is convex if the line segment between any two points in C lies in C, i.e., if for any $x, y \in C$ and any α with $0 \leq \alpha \leq 1$, we have: $\alpha x + (1-\alpha)y \in C$ [16].*

Definition 2.3. *A function $h : \mathbb{R}^N \to \mathbb{R}$ is convex if its domain D is a convex set, and for all $\mathbf{x}, \mathbf{y} \in D$ and α with $0 \leq \alpha \leq 1$, we have: $h(\alpha x + (1-\alpha)y) \leq \alpha h(x) + (1-\alpha)h(y)$ [16].*

2.1 Convex Markov Decision Process (CMDP)

Definition 2.4. *A CMDP is a tuple $\mathcal{M}_\mathcal{C} = (S, S_0, A, \Omega, \mathcal{F}, \mathcal{A}, \mathcal{X}, L)$, where S is a finite set of states of cardinality $N = |S|$, S_0 is the set of initial states, A is a finite set of actions ($M = |A|$), Ω is a finite set of atomic propositions, $\mathcal{F}$ is a finite set of convex sets of transition PDs, $\mathcal{A} : S \to 2^A$ is a function that maps each state to the set of actions available at that state, $\mathcal{X} = S \times A \to \mathcal{F}$ is a function that associates* *to state s and action a the corresponding convex set $\mathcal{F}_s^a \in \mathcal{F}$ of transition PDs, and $L : S \to 2^{\Omega}$ is a labeling function.*

The set $\mathcal{F}_s^a = Dist_s^a(S)$ represents the uncertainty in defining a transition distribution for $\mathcal{M}_{\mathcal{C}}$ given state s and action a. We call $\mathbf{f}_s^a \in \mathcal{F}_s^a$ an observation of this uncertainty. Also, $\mathbf{f}_s^a \in \mathbb{R}^N$ and we can collect the vectors $\mathbf{f}_s^a, \forall s \in S$ into an observed transition matrix $F^a \in \mathbb{R}^{N \times N}$. Abusing terminology, we call $\mathcal{F}^a$ the uncertainty set of the transition matrices, and $F^a \in \mathcal{F}^a$. $\mathcal{F}_s^a$ is interpreted as the row of $\mathcal{F}^a$ corresponding to state s. Finally, $f_{s_i s_j}^a = \mathbf{f}_{s_i}^a[j]$ is the observed probability of transitioning from s_i to s_j when action a is selected.

A transition between state s to state s' in a CMDP occurs in three steps. First, an action $a \in \mathcal{A}(s)$ is chosen. The selection of a is nondeterministic. Secondly, an observed PD $\mathbf{f}_s^a \in \mathcal{F}_s^a$ is chosen. The selection of $\mathbf{f}_s^a$ models uncertainty in the transition. Lastly, a successor state s' is chosen randomly, according to the transition PD $\mathbf{f}_s^a$.

A path π in $\mathcal{M}_{\mathcal{C}}$ is a finite or infinite sequence of the form $s_0 \xrightarrow{f_{s_0 s_1}^{a_0}} s_1 \xrightarrow{f_{s_1 s_2}^{a_1}}, \cdots,$ where $s_i \in S$, $a_i \in \mathcal{A}(s_i)$ and $f_{s_i, s_{i+1}}^{a_i} > 0 \ \forall i \geq 0$. We indicate with Π_{fin} (Π_{inf}) the set of all finite (infinite) paths of $\mathcal{M}_{\mathcal{C}}$. $\pi[i]$ is the i^{th} state along the path and, for finite paths, $last(\pi)$ is the last state visited in $\pi \in \Pi_{fin}$. $\Pi_s = \{\pi \mid \pi[0] = s\}$ is the set of paths starting in state s.

Although we consider CMDPs as the underlying model of the system, the proposed techniques can be extended also to Convex Markov Chains (CMCs), which can be seen as CMDPs with a single action, i.e., $M = 1$.

To model uncertainty in state transitions, we make the following assumptions:

Assumption 2.1. *$\mathcal{F}^a$ can be factored as the Cartesian product of its rows, i.e., its rows are uncorrelated. Formally, for every $a \in A$, $\mathcal{F}^a = \mathcal{F}_{s_0}^a \times \cdots \times \mathcal{F}_{s_{N-1}}^a$. In [7] this assumption is referred to as* rectangular uncertainty.

Assumption 2.2. *If the probability of a transition is zero (non-zero) for at least one PD in the uncertainty set, then it is zero (non-zero) for all PDs.*
Formally, $\exists \mathbf{f}_s^a \in \mathcal{F}_s^a : f_{ss'}^a = (\neq) 0 \implies \forall \mathbf{f}_s^a \in \mathcal{F}_s^a : f_{ss'}^a = (\neq) 0$. The assumption guarantees the correctness of the preprocessing verification routines used later in this work, which rely on reachability of the states of the MDP underlying graph.

We determine the size $\mathcal{R}$ of the CMDP $\mathcal{M}_{\mathcal{C}}$ as follows. $\mathcal{M}_{\mathcal{C}}$ has N states, $O(M)$ actions per state and $O(N^2)$ transitions for each action. Let D_s^a denote the number of constraints required to express the rectangular uncertainty set $\mathcal{F}_s^a$ (e.g. $D_s^a =$

$O(2N)$ for the interval model, to express the upper and lower bounds of the transition probabilities from state s to all states $s' \in S$), and $D = \underset{s \in S, a \in A}{max} D_s^a$. The overall size of $\mathcal{M}_\mathcal{C}$ is thus $\mathcal{R} = O(N^2 M + NMD)$.

In order to analyze *quantitative* properties of CMDPs, we need a probability space over infinite paths [17]. However, a probability space can only be constructed once nondeterminism and uncertainty have been resolved. We call each possible resolution of nondeterminism an *adversary*, which chooses an action in each state of $\mathcal{M}_\mathcal{C}$.

Definition 2.5. Adversary. *A randomized adversary for $\mathcal{M}_\mathcal{C}$ is a function $\alpha = \Pi_{fin} \times A \to [0,1]$, with $\sum_{\mathcal{A}(last(\pi))} \alpha(\pi, a) = 1$, and $a \in \mathcal{A}(last(\pi))$ if $\alpha(\pi, a) > 0$. We call Adv the set of all adversaries α of $\mathcal{M}_\mathcal{C}$.*

Conversely, we call a *nature* each possible resolution of uncertainty, i.e., a nature chooses a transition PD for each state and action of $\mathcal{M}_\mathcal{C}$.

Definition 2.6. Nature. *Given action $a \in A$, a randomized nature is the function $\eta^a : \Pi_{fin} \times Dist(S) \to [0,1]$ with $\int_{\mathcal{F}^a_{last(\pi)}} \eta^a(\pi, \mathbf{f}^a_s) = 1$, and $\mathbf{f}^a_s \in \mathcal{F}^a_{last(\pi)}$ if $\eta^a(\pi, \mathbf{f}^a_s) > 0$. We call Nat the set of all natures η^a of $\mathcal{M}_\mathcal{C}$.*

An adversary α (nature η^a) is memoryless if it depends only on $last(\pi)$. Also, α (η^a) is deterministic if $\alpha(\pi, a) = 1$ for some $a \in \mathcal{A}(last(\pi))$ ($\eta^a(\pi, f^a_s) = 1$ for some $\mathbf{f}^a_s \in \mathcal{F}^a_{last(\pi)}$).

2.2 Models of Uncertainty

We only consider CMDPs whose transition PDs lie in uncertainty sets that satisfy Assumption 5.1 (introduced later for ease of presentation). This assumption holds for all the uncertainty models analyzed in [7]. We report results for the interval, likelihood, ellipsoidal and entropy models.

Interval Model. Intervals commonly describe uncertainty in transition matrices:

$$\mathcal{F}^a_s = \{\mathbf{f}^a_s \in \mathbb{R}^N \mid \mathbf{0} \leq \underline{\mathbf{f}}^a_s \leq \mathbf{f}^a_s \leq \overline{\mathbf{f}}^a_s \leq \mathbf{1}, \mathbf{1}^T \mathbf{f}^a_s = 1\} \tag{1}$$

where $\underline{\mathbf{f}}^a_s, \overline{\mathbf{f}}^a_s \in \mathbb{R}^N$ are the element-wise lower and upper bounds of $\mathbf{f}$. This model is suitable when the transition matrix components are individually estimated by statistical data.

Likelihood Model. This model is appropriate when the transition probabilities are determined experimentally. The transition frequencies associated to action $a \in A$ are collected in matrix H^a. Uncertainty in each row of H^a can be described by the likelihood region [18]:

$$\mathcal{F}_s^a = \{\mathbf{f}_s^a \in \mathbb{R}^N \mid \mathbf{f}_s^a \geq \mathbf{0}, \mathbf{1}^T\mathbf{f}_s^a = 1, \textstyle\sum_{s'} h_{ss'}^a \log(f_{ss'}^a) \geq \beta_s^a\} \tag{2}$$

where $\beta_s^a < \beta_{s,max}^a = \sum_{s'} h_{ss'}^a \log(h_{ss'}^a)$ represents the uncertainty level. Likelihood regions are less conservative uncertainty representations than intervals, which arise from projections of the uncertainty region onto each row component.

Ellipsoidal Model. Ellipsoidal models can be seen as a second-order approximation of the likelihood model [7]. Formally:

$$\mathcal{F}_s^a = \{\mathbf{f}_s^a \in \mathbb{R}^N \mid \mathbf{f}_s^a \geq \mathbf{0}, \mathbf{1}^T\mathbf{f}_s^a = 1, \|R_s^a\left(\mathbf{f}_s^a - \mathbf{h}_s^a\right)\|_2 \leq 1, R_s^a \succ 0\} \tag{3}$$

where matrix R_s^a represents an ellipsoidal approximation of the likelihood Region (2).

Entropy Model The entropy model of uncertainty can be viewed as a variation of the likelihood model. In the likelihood setting we bound the divergence from an empirically extracted distribution, whereas in the entropy setting we bound the divergence from a reference analytical distribution q [7]. We will thus consider sets:

$$\mathcal{F}_s^a = \{\mathbf{f}_s^a \in \mathbb{R}^N \mid \mathbf{f}_s^a \geq \mathbf{0}, \mathbf{1}^T\mathbf{f}_s^a = 1, \textstyle\sum_{s'} f_{ss'}^a \log\left(\frac{f_{ss'}^a}{q_{ss'}^a}\right) \leq \beta_s^a\} \tag{4}$$

Remark 2.1. Each set $\mathcal{F}_s^a$ within the same CMDP can be expressed with a different uncertainty model to represent different sources of uncertainty.

To illustrate our results, we will use the IMDP $\mathcal{M}_\mathcal{C}$ in Figure 1, with $S = \{s_0 \cdots s_3\}$, $S_0 = \{s_0\}$, $A = \{a, b\}$, $\Omega = \{\omega, \vartheta\}$, $\mathcal{A} : \{s_0, s_1, s_2\} \rightarrow \{a\}\ ; \{s_3\} \rightarrow \{a, b\}$, $L : \{s_0, s_3\} \rightarrow \vartheta\ ; \{s_2\} \rightarrow \omega$. The uncertainty intervals are shown next to each transition. For example,

$\mathcal{F}_{s_0}^a = \{\mathbf{f}_{s_0}^a \in \mathbb{R}^N \mid [0, 0.6, 0.2, 0] \leq \mathbf{f}_{s_0}^a \leq [0, 0.8, 0.5, 0], \sum_{s' \in S} f_{ss'}^a = 1\}$.

2.3 Probabilistic Computation Tree Logic (PCTL)

We use PCTL, a probabilistic logic derived from CTL which includes a probabilistic operator P [5], to express properties of CMDPs. The syntax of this logic is defined

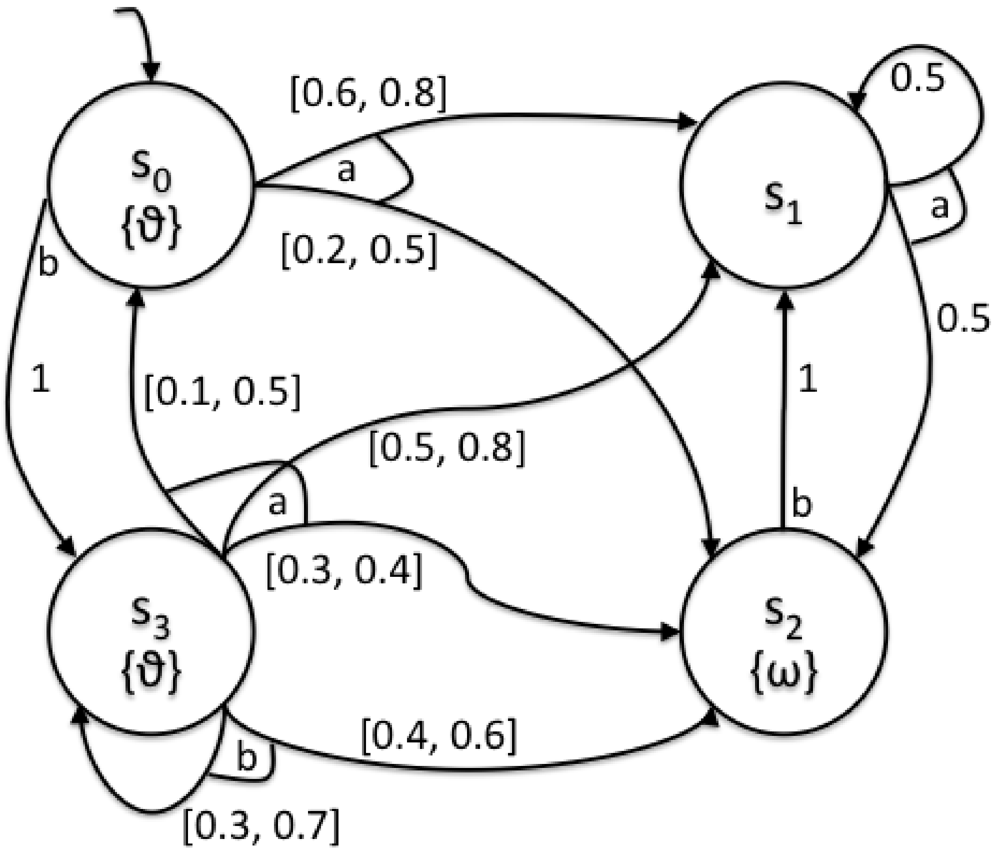

Figure 1: Simple Interval-MDP used to exemplify the operation of the proposed algorithms.

as follows:

$$\phi ::= True \mid \omega \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid P_{\bowtie p}[\psi] \qquad \text{state formulas}$$
$$\psi ::= \mathcal{X}\phi \mid \phi_1\,\mathcal{U}^{\leq k}\phi_2 \mid \phi_1\,\mathcal{U}\phi_2 \qquad \text{path formulas}$$

where $\omega \in \Omega$ is an atomic proposition, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$ and $k \in \mathbb{N}$.

Path formulas ψ use the *Next* $(\mathcal{X})$, *Bounded Until* $\left(\mathcal{U}^{\leq k}\right)$ and *Unbounded Until* $(\mathcal{U})$ operators. These formulas are evaluated over paths and only allowed as parameters to the $P_{\bowtie p}[\psi]$ operator. The size $\mathcal{Q}$ of a PCTL formula is defined as the number of Boolean connectives plus the number of temporal operators in the formula. For the *Bounded Until* operator, we denote separately the maximum time bound that appears in the formula as k_{max}. Probabilistic statements about MDPs typically involve

Table 2: PCTL semantics for Convex-MDPs

$s \models \text{True}$	
$s \models \omega$	iff $\omega \in L(s)$
$s \models \neg\phi$	iff $s \not\models \phi$
$s \models \phi_1 \wedge \phi_2$	iff $s \models \phi_1 \wedge s \models \phi_2$
$s \models P_{\lhd p}[\psi]$	iff $P_s^{max}(\{\pi \in \Pi_s \mid \pi \models \psi\}) \lhd p$
$s \models P_{\rhd p}[\psi]$	iff $P_s^{min}(\{\pi \in \Pi_s \mid \pi \models \psi\}) \rhd p$
$\pi \models \mathcal{X}\phi$	iff $\pi[1] \models \phi$
$\pi \models \phi_1\, \mathcal{U}^{\leq k}\phi_2$	iff $\exists i \leq k \mid \pi[i] \models \phi_2 \wedge \forall j < i\ \pi[j] \models \phi_1$
$\pi \models \phi_1\, \mathcal{U}\phi_2$	iff $\exists k \geq 0 \mid \pi \models \phi_1\, \mathcal{U}^{\leq k}\phi_2$

universal quantification over adversaries $\alpha \in Adv$. With uncertainties, for each action a selected by adversary α, we will further quantify across nature $\eta^a \in Nat$ to compute the worst case condition within the action range of η^a, i.e., the uncertainty set $\mathcal{F}_s^a$. We define $P_s(\alpha, \eta^a)[\psi] \triangleq Prob(\{\pi \in \Pi_s(\alpha, \eta^a) \mid \pi \models \psi\})$ the probability of taking a path $\pi \in \Pi_s$ that satisfies ψ under adversary α and nature η^a. If α and η^a are Markov deterministic in state s, we write $P_s(a, \mathbf{f}_s^a)$, where a and $\mathbf{f}_s^a$ are the action and resolution of uncertainty that are deterministically chosen at each execution step by α and η^a. $P_s^{max}[\psi]$ ($P_s^{min}[\psi]$) denote the maximum (minimum) probability $P_s(\alpha, \eta^a)[\psi]$ across all adversaries $\alpha \in Adv$ and natures $\eta^a \in Nat$, and the vectors $\mathbf{P}^{max}[\psi], \mathbf{P}^{min}[\psi] \in \mathbb{R}^N$ collect these probabilities $\forall s \in S$. The semantics of the logic is reported in Table 2, where we write $\models$ instead of $\models_{Adv,Nat}$ for simplicity.

For ease of computation, we would like to restrict our attention to memoryless and deterministic adversaries and natures to compute *quantitative* probabilities, i.e., solve problems:

$$P_s^{max}[\psi] = \max_{a \in \mathcal{A}(s)} \max_{\mathbf{f}_s^a \in \mathcal{F}_s^a} P_s(a, \mathbf{f}_s^a)[\psi] \overset{?}{\leq} p \tag{5}$$

or

$$P_s^{min}[\psi] = \min_{a \in \mathcal{A}(s)} \min_{\mathbf{f}_s^a \in \mathcal{F}_s^a} P_s(a, \mathbf{f}_s^a)[\psi] \overset{?}{\geq} p \tag{6}$$

We extend a result from [19] to prove that this is possible.

Proposition 2.1. *Given a CMDP $\mathcal{M}_\mathcal{C}$ and a target state $s_t \in S$, there always exist deterministic and memoryless adversaries and natures for $\mathcal{M}_\mathcal{C}$ that achieve the maximum (minimum) probabilities of reaching s_t, if A is finite and the in-*

ner optimization in Problem (6) always attains its optimum $\sigma_s^*(a)$ over the sets $\mathcal{F}_s^a, \forall s \in S, \forall a \in \mathcal{A}(s)$, i.e., there exists a finite feasible $\mathbf{f}_s^a \in \mathcal{F}_s^a$ such that $P_s(a, \mathbf{f}_s^a)[\psi] = \sigma_s^*(a)$.

Sketch of proof. The proof is divided into two parts. First, we prove the existence of an adversary and a nature that achieve the maximum (minimum) probabilities of reaching t_s, using Banach fixed-point theorem [19]. Second, we prove that at least one such adversary and nature is memoryless and deterministic. The proof extends the one in Puterman [19], Theorem $6.2.10$. We need to prove that Problem (6) always attains the maximum (minimum) over the feasibility sets $\mathcal{F}_s^a$, i.e., $\forall s \in S, \forall a \in \mathcal{A}(s), \exists \mathbf{f}_s^a \in \mathcal{F}_s^a : ||\mathbf{f}_s^a||_2 < \infty, P_s(a, \mathbf{f}_s^a)[\psi] = \sigma_s^*(a)$. This is indeed true for all the convex sets $\mathcal{F}_s^a$ considered in this thesis. The interval and ellipsoidal models result in *compact* sets $\mathcal{F}_s^a$ on which Weierstrass theorem holds. For the likelihood model we use the notion of *consistency*, which guarantees the existence and uniqueness of a point in $\mathcal{F}_s^a$ where the optimum is attained. □

The verification of a PCTL state formula ϕ can be viewed as a decision problem. The verification algorithm V needs to determine whether a state $s \in S_0$ is (or is not) contained in the set $Sat(\phi) = \{s \in S \mid s \models \phi\}$. We can thus define the following properties for V:

Definition 2.7. Soundness *(*Completeness*). Algorithm V is sound (complete) if:*

$$s \in Sat_V(\phi) \Rightarrow s \in Sat(\phi) \qquad (s \notin Sat_V(\phi) \Rightarrow s \notin Sat(\phi))$$

where $Sat_V(\phi)$ is the computed satisfaction set, while $Sat(\phi)$ is the actual satisfaction set.

Algorithms to verify non-probabilistic formulas are sound and complete, because they are based on reachability analysis over the finite number of states of $\mathcal{M}_\mathcal{C}$ [20]. Conversely, we will show in Section 5 that algorithms to verify probabilistic formulas $\phi = P_{\bowtie p}[\psi]$ in the presence of uncertainties require to solve convex optimization problems over the set $\mathbb{R}$ of the real numbers. Optima of these problems can be arbitrary real numbers, so, in general, they can be computed only to within a desired accuracy ϵ_d. We consider an algorithm to be sound and complete if the error in determining the satisfaction probabilities of ϕ is bounded by such a parameter ϵ_d, since the returned result will still be accurate enough in most settings.

Chapter 3

Related Work

In this chapter, we give an overview of previous work presented in the literature both in the area of formal verification and robust control. We begin with a brief review of model-checking techniques and their theoretical complexity. This analysis clarifies the improvement in theoretical complexity described in this thesis. We then present model-checking tools proposed in the literature and conclude with references to works focusing on the problem of devising robust control strategies for MDPs with uncertainties.

Automatic verification techniques such as model checking [21] are ways of establishing formal guarantees for a system with respect to given logical specifications. Traditionally, model checking has been focused on developing methods for analyzing *qualitative* properties of system models, e.g. checking that a communication protocol never deadlocks. Many applications however often involve stochastic behaviors due to interaction with physical processes (e.g. failure of an unreliable component) or built-in randomization (e.g. random back-off schemes in IEEE 802.11). These applications need *quantitative* analysis [4] in order to answer questions such as "what is the probability that a request will be served within a given time limit?". Stochastic models like Discrete-Time Markov Chains (DTMCs) [2] and Markov Decision Processes (MDPs) [3] have been used to formally represent systems that exhibit random or probabilistic behaviors. These systems need *quantitative* analysis [4] to answer questions such as "what is the probability that a request will be eventually served?". Properties of these systems can be expressed and analyzed using logics such as Probabilistic Computation Tree Logic (PCTL) [5] — a probabilistic logic derived from CTL which includes a probabilistic operator P — as well as techniques for probabilistic model checking [6]. These methods often rely on deriving a probabilistic model of the underlying process, hence the formal guarantees they provide are only as good as the estimation. In a real setting, these estimations are affected by uncertainties due for example to unmodeled dynamics, measurement errors or approximation of the real system by mathematical models.

Interval-valued Discrete-Time Markov Chains (IDTMCs) have been introduced to capture modeling uncertainties [8]. IDTMCs are DTMC models where each transition probability is assumed to lie within a compact range. Two semantic interpretations have been proposed for these models [9]: Uncertain Markov Chains (UMCs) and Interval Markov Decision Processes (IMDPs). An UMC is interpreted as a family of (possibly uncountably many) DTMCs, where each member of the family is a DTMC whose transition probabilities lie within the interval range given in the UMC. In IMDPs, the uncertainty is resolved through non-determinism. Each time a state is visited, a transition distribution within the interval constraints is adversarially picked, and a probabilistic step is taken accordingly. Thus, IMDPs allow modeling a non-deterministic choice made from a set of (possibly uncountably many) choices. In this thesis, we do not consider UMCs and focus on IMDPs. Moreover, while all previous work focused only on interval models of uncertainty, we also incorporate more expressive models such as the ellipsoidal, likelihood and entropy models [7].

An upper-bound on the complexity of model checking PCTL properties on IMDPs was previously shown to be PSPACE [9], and the result was later improved to co-NP [10]. These results rely on the construction of an equivalent MDP that encodes all behaviors of the IMDP. For each state in the new MDP, the set of transition probabilities is equal to the Basic Feasible Solutions (BFS) of the set of inequalities specifying the transition probabilities of the IMDP. Since in the worst case the number of BFS is exponential in the number of states in the IMDP, the equivalent MDP can have size exponential in the size of the IMDP.

In this work, we improve the previously best-known complexity result of co-NP in [10] to PTIME, for the fragment of PCTL without $\mathcal{U}^{\leq k}$. We also characterize the complexity of our algorithm for the full PCTL syntax [5] based on the size $\mathcal{R}$ of the IMDP model, the size $\mathcal{Q}$ of the PCTL formula, and the maximum bound k_{max} in the $\mathcal{U}^{\leq k}$ operators. Our algorithm runs in $O(poly(\mathcal{R}) \times \mathcal{Q} \times k_{max})$ time, which is pseudo-polynomial in k_{max} (i.e. polynomial if k_{max} is counted in its unary representation and exponential if k_{max} is counted in its binary representation). On the other hand, classical PCTL model checking for DTMCs [5] runs in time polynomial in k_{max} counted in its binary representation. The difference stems from the computation of the set $Sat\left(P_{\bowtie p}\left[\phi_1\,\mathcal{U}^{\leq k}\phi_2\right]\right)$. For (certain) MDPs, this computation involves raising the transition matrices $F^a, \forall a \in A$ to the k^{th} power, to model the evolution of the system in k steps. With uncertainties, we cannot do matrix exponentiation, because $F^a \in \mathcal{F}^a$ might change at each step. However, matrix exponentiation is seldom used in practice because the transition matrix of an MDP is often sparse and repeatedly squaring it causes fill-ins [20]. Moreover, both $\mathcal{Q}$ and k_{max} are typically small in practical applications [13, 22, 23], so the dominant factor for runtime is the size of the model $\mathcal{R}$. Hence, our algorithm is favorable in terms of scalability of the verification problem. We note that the complexity results of [9] and [10] can be extended to the PCTL with $\mathcal{U}^{\leq k}$.

In parallel to the formalization of the aforementioned theoretical results, the community has also worked on the development of tools for the model checking of probabilistic systems. To list a few, we mention INFAMY [24], MRMC [25], ETMCC [26], VESTA [27], Ymer [28] and the PRISM Model Checker [6]. At the time of writing, the PRISM Model Checker appears to be at the frontier in this effort. The tool has been used to analyze a multitude of applications, from communication protocols and biological pathways to security problems. PRISM is capable of analyzing several different probabilistic models (e.g. discrete and continuous-time Markov Chains (DTMCs/CTMCs), Markov Decision Processes (MDPs), Probabilistic Timed Automata (PTA)) and properties expressed in a variety of specification languages (e.g. Probabilistic Computation Tree Logic, Linear Temporal Logic). Along the years, the tool has been used to verify and analyze properties of a heterogeneous set of systems ranging from biochemical reactions and power-management units to security protocols and many others. By further considering *uncertainties* in the probabilistic transitions of the MDP for model checking PCTL specifications, our work extends the capabilities of these tools.

Convex uncertainty models [7] similar to the ones analyzed in this thesis have been considered recently in the robust control literature, where the problem is to synthesize a robust optimal controller for an MDP in order to satisfy a Linear Temporal Logic (LTL) specification where only one probabilistic operator is allowed [29]. Their technique first converts the LTL specification to a Rabin automaton (which is worst-case doubly exponential in the size of the LTL formula), and composes it with the MDP. Robust dynamic programming is then used to solve for the optimal control policy similar to [7]. We consider PCTL, and allow nested probability operators. Additionally, our algorithm is polynomial both in the size of the model and of the formula, leveraging results on strong duality for convex programs.

Recently, there has also been work that aims to provide robustness to PCTL model checking based on the notion of Approximate Probabilistic Bisimulation (APB) [30]. In that paper, the existence of an APB to a Labeled Markov Chain (LMC) with precision ϵ is proven to imply the preservation of ϵ-robust PCTL formulas in the LMC. The notion of APB is tailored to finite-precision approximation of a numerical model. Our work is different because we aim to check PCTL formulas of MDPs whose transition probabilities are affected by uncertainties due to estimation errors or imperfect information about the environment. Further, our goal is different in that we aim to expose the effect of uncertainties (small perturbations) in the process behavior on the satisfaction of the property that we are verifying.

Chapter 4

Probabilistic Model Checking with Uncertainties

In this chapter, we formally define the verification problem analyzed in this thesis, and give an overview of the proposed approach to solve it.

PCTL model checking with uncertainties. **Given** a Markov Decision Process model with convex uncertainties $\mathcal{M}_{\mathcal{C}}$ of size $\mathcal{R}$ and a PCTL formula ϕ of size $\mathcal{Q}$ over a set of atomic propositions Ω, **verify** ϕ over the uncertainty sets $\mathcal{F}_s^a \in \mathcal{F}$ of $\mathcal{M}_{\mathcal{C}}$.

As in verification of CTL [31], the algorithm traverses bottom-up the parse tree for ϕ, recursively computing the set $Sat(\phi')$ of states satisfying each sub-formula ϕ'. At the end of the traversal, the algorithm computes the set of states satisfying ϕ and it determines if $s \models \phi$ by checking if $s \in Sat\,(\phi)$. For the non-probabilistic PCTL operators, the satisfying states are computed as: $Sat\,(True) = S$, $Sat(\omega) = \{s \in S \mid \omega \in L(s)\}$, $Sat(\neg\phi) = S \backslash Sat(\phi)$ and $Sat(\phi_1 \wedge \phi_2) = Sat(\phi_1) \cap Sat(\phi_2)$. For the probabilistic operator $P \bowtie [\psi]$, we compute:

$$Sat\,(P_{\lhd p}\,[\psi]) = \{s \in S \mid P_s^{max}(\psi) \lhd p\} \tag{7}$$

$$Sat\,(P_{\rhd p}\,[\psi]) = \left\{s \in S \mid P_s^{min}(\psi) \rhd p\right\} \tag{8}$$

In this thesis, we propose polynomial-time routines to compute Sets (7) - (8) for MDPs whose transition matrices F^a are only known to lie within convex uncertainty sets $\mathcal{F}^a$, $\forall a \in A$.

Using Proposition 2.1, the proposed routines encode the transitions of $\mathcal{M}_{\mathcal{C}}$ under the sets of deterministic and memoryless adversaries and natures into convex programs and solve them. From the returned solution, it is then possible to determine the *quantitative* satisfaction probabilities $P_s^{max}[\psi]$ (or $P_s^{min}[\psi]$) $\forall s \in S$, which get compared in linear time to the threshold p to compute the set $Sat\,(P_{\bowtie p}\,[\psi])$. To prove the polynomial-time complexity of the model-checking algorithm, we use the following key result from convex theory [32].

Proposition 4.1. *Given the convex program:*

$$\begin{aligned} &\underset{\mathbf{x}}{\mathit{min}}\ f_0(\mathbf{x}) \\ &\mathit{s.t.}\ f_i(\mathbf{x}) \leq 0 \qquad\qquad\qquad i = 1, \cdots, m \end{aligned}$$

with $\mathbf{x} \in \mathbb{R}^n$ *and* $f_i, i = 0, \cdots, m$ *convex functions, the optimum* σ^* *can be found to within* $\pm\epsilon_d$ *in time complexity that is polynomial in the size of the problem* (n, m) *and* $\log(1/\epsilon_d)$.

We are now ready to state the main contribution of this thesis:

Theorem 4.1. Complexity of PCTL model-checking for CMDPs.

1. *The problem of verifying if a CMDP* $\mathcal{M}_\mathcal{C}$ *of size* $\mathcal{R}$ *satisfies a PCTL formula* ϕ *without* $\mathcal{U}^{\leq k}$ *is in PTIME.*
2. *A formula* ϕ' *with* $\mathcal{U}^{\leq k}$ *can be verified with time complexity* $O\left(poly(\mathcal{R}) \times \mathcal{Q}' \times k_{max}\right)$, *i.e., pseudo-polynomial in the maximum time bound* k_{max} *of* $\mathcal{U}^{\leq k}$.

Sketch of proof. The proof is constructive. Our verification algorithm parses ϕ in time linear in the size $\mathcal{Q}$ of ϕ [31], computing the satisfiability set of each operator in ϕ. For the non-probabilistic operators, satisfiability sets can be computed in time polynomial in $\mathcal{R}$ using set operations, i.e., set inclusion, complementation and intersection. For the probabilistic operator, we leverage Proposition 4.1 and prove that the proposed verification routines: 1) solve a number of convex problems polynomial in $\mathcal{R}$; 2) generate these convex programs in time polynomial in $\mathcal{R}$. The correctness and time-complexity for formulas involving *Next* and *Unbounded Until* operators are formalized in Lemma 5.1 and 5.2 (Section 5.1 and 5.2). It thus follows that the overall algorithm runs in time polynomial in $\mathcal{R}$ and in the size of ϕ. Finally, Lemma 5.3 formalizes the results related to the *Bounded Until* operator. $\square$

Chapter 5
Verification Routines

In this chapter, we give details about the routines for the verification of the temporal operators allowed in PCTL. In particular, we analyze the Next, Unbounded Until *and* Bounded Until *operators. For each operator, we first present the algorithmic procedure to determine the states in the Convex-MDP satisfying the PCTL formula ϕ. We then prove the soundness and completeness of the procedure and derive its algorithmic complexity.*

We detail the routines used to verify the probabilistic operator P. We only consider properties in the form $\phi = P_{\triangleleft p}[\psi]$, but the results can trivially be extended to $\phi = P_{\triangleright p}[\psi]$ [33] by replacing "max" with "min" in the optimization problems below.

5.1 *Next* Operator

We present the routine to verify property $\phi = P_{\triangleleft p}[\mathcal{X}\phi_1]$ on a CMDP of size $\mathcal{R}$. First, the set $S^{yes} = Sat(\phi_1)$ of all states satisfying ϕ_1 is computed. Second, for each state, the algorithm computes the probabilities defined in Equation (6) by solving the problem:

$$P_s^{max}[\mathcal{X}\phi_1] = \max_{a \in \mathcal{A}(s)} \max_{\mathbf{f}_s^a \in \mathcal{F}_s^a} \sum_{s' \in S^{yes}} f_{ss'}^a \tag{9}$$

In Problem (9), the inner $\max$ is a convex program since $\mathcal{F}_s^a$ is convex. We note that the sets $\mathcal{F}_s^a$ can be expressed with models of uncertainty different from one another, since each optimization problem is independent from the others. Finally, the computed probabilities are compared to the threshold p to select the states that satisfy ϕ.

Lemma 5.1. *The routine to verify the* Next *operator is sound, complete and guaranteed to terminate with algorithmic complexity that is polynomial in the size $\mathcal{R}$ of $\mathcal{M}_\mathcal{C}$.*

Proof. From Problem (9) we see that there is one "inner" convex program for each state $s \in S$ and action $a \in \mathcal{A}(s)$, for a total of at most MN problems. Each problem has at most N unknowns, representing the probability of transitioning from state s

to state s' for $s' \in S^{yes}$. It has $N+1$ constraints to guarantee that the solution lies in the probability simplex, and D_s^a constraints to enforce the solution to be within the uncertainty set $\mathcal{F}_s^a$. According to the definition in Section 2.1, the total number of constraints is linear in $\mathcal{R}$. Using Proposition 4.1, each inner problem is solved in time polynomial in $\mathcal{R}$. Soundness and completeness also follow directly from Proposition 4.1, which states that the optimum of Problem (9) can be found to within the desired accuracy $\pm\epsilon_d$ in time linear in $\log(1/\epsilon_d)$. □

We verify $\phi = P_{\leq 0.4}[\mathcal{X}\omega]$ in the example in Figure 1. Trivially, $S^{yes} = \{s_2\}$. Setting up the inner problem for state s_0 and action a, we get:

$$P_{s_0}^{a,max} = \max_{f_{01}, f_{02}} f_{02}$$
$$\text{s.t. } 0.6 \leq f_{01} \leq 0.8;\ 0.2 \leq f_{02} \leq 0.5;\ f_{01} + f_{02} = 1$$

with solution $P_{s_0}^{a,max}[\mathcal{X}\omega] = 0.4$. Repeating $\forall a \in A$,$\forall s \in S$, we get $\mathbf{P}^{max}[\mathcal{X}\omega] = [0.4, 0.5, 0, 0.6]$, so $Sat(\phi)$=$\{s_0, s_2\}$.

5.2 *Unbounded Until* Operator

We present the routine to verify $\phi = P_{\triangleleft p}[\phi_1 \mathcal{U} \phi_2]$ on a CMDP of size $\mathcal{R}$. First, the sets $S^{yes} \overset{def}{=} Sat\,(P_{\geq 1}[\phi_1 \mathcal{U} \phi_2])$, $S^{no} \overset{def}{=} Sat\,(P_{\leq 0}[\phi_1 \mathcal{U} \phi_2])$ and $S^{?} = S \setminus (S^{no} \cup S^{yes})$ are precomputed in time polynomial in $\mathcal{R}$ using conventional reachability routines over the CMDP underlying graph [20]. Second, the maximum probability to satisfy ϕ, as introduced in Equation (6), is computed for all states $s \in S$ at the same time using the *Convex Programming* procedure described next. Finally, the computed probabilities are compared to the threshold p.

Convex Programming Procedure (CP). We start from the classical LP formulation to solve the problem without the presence of uncertainty [20]:

$$\begin{aligned} &\min_{\mathbf{x}} \mathbf{x}^{\mathbf{T}}\mathbf{1} \\ &\text{s.t. } x_s = 0;\ x_s = 1; && \forall s \in S^{no};\ s \in S^{yes}; \\ &\quad x_s \geq \mathbf{x}^{\mathbf{T}}\mathbf{f}_{\mathbf{s}}^{\mathbf{a}} && \forall s \in S^{?}, \forall a \in \mathcal{A}(s) \end{aligned} \tag{10}$$

where $\mathbf{P}^{max}[\phi_1 \mathcal{U} \phi_2] = \mathbf{x}^*$ is computed solving only one LP. Problem (10) has N unknowns and $N - Q + MQ$ constraints, where $Q = |S^{?}| = O(N)$, so its size is polynomial in $\mathcal{R}$.

Proposition 2.1 allows us to rewrite Problem (10) in the uncertain scenario as:

$$\begin{aligned}
&\min_{\mathbf{x}} \mathbf{x^T 1} && \\
&\text{s.t. } x_s = 0;\ x_s = 1; && \forall s \in S^{no};\ \forall s \in S^{yes}; \\
&\quad x_s \geq \max_{\mathbf{f_s^a} \in \mathcal{F}_s^a} (\mathbf{x^T f_s^a}) && \forall s \in S^?, \forall a \in \mathcal{A}(s)
\end{aligned} \tag{11}$$

i.e., we maximize the lower bound on x_s across the nature action range. The decision variable of the inner problem is $\mathbf{f}_s^a$ and its optimal value $\sigma^*(\mathbf{x})$ is parametrized in the outer problem decision variable $\mathbf{x}$. Problem (11) can be written in convex form for an arbitrary uncertainty model by replacing the last constraint with a set of constraints, one for each point in $\mathcal{F}_s^a$. However, this approach results in infinite constraints if the set $\mathcal{F}_s^a$ contains infinitely many points, as in the cases considered in this work, making the problem unsolvable. We solve this difficulty using duality, which allows to rewrite Problem (11) with a number of additional constraints only polynomial in the size of the CMDP. For each state $s \in S^?$ and action $a \in \mathcal{A}(s)$, we replace the primal inner problem in the outer Problem (11) with its dual:

$$\sigma_s^a(\mathbf{x}) = \max_{\mathbf{f}_s^a \in \mathcal{F}_s^a} \mathbf{x}^T \mathbf{f}_s^a \qquad \Rightarrow \qquad d_s^a(\mathbf{x}) = \min_{\lambda_\mathbf{s}^\mathbf{a} \in \mathcal{D}_s^a} g(\lambda_\mathbf{s}^\mathbf{a}, \mathbf{x}) \tag{12}$$

where $\lambda_\mathbf{s}^\mathbf{a}$ is the (vector) Lagrange multiplier and $\mathcal{D}_s^a$ is the feasibility set of the dual problem. In the dual, the decision variable is $\lambda_\mathbf{s}^\mathbf{a}$ and its optimal value $d_s^a(\mathbf{x})$ is again parametrized in the outer problem decision variable $\mathbf{x}$. The dual function $g(\lambda_s^a, \mathbf{x})$ and the set $\mathcal{D}_s^a$ are convex by construction in λ_s^a for arbitrary uncertainty models, so the dual problem is convex. Further, since also the primal problem is convex, strong duality holds, i.e., $\sigma_s^a = d_s^a, \forall \mathbf{x} \in \mathbb{R}^N$, because the primal problem satisfies Slater's condition [16] for any non-trivial uncertainty set $\mathcal{F}_s^a$. Any dual solution overestimates the primal solution. When substituting the primals with the duals in Problem (11), we can drop the inner optimization operators because the outer optimization operator will nevertheless aim to find the least overestimates, i.e., the dual solutions $d_s^a, \forall s \in S, a \in \mathcal{A}(s)$, to minimize its cost function. We get the CP formulation:

$$\begin{aligned}
&\min_{\mathbf{x}} \mathbf{x^T 1} \\
&\text{s.t. } x_s = 0;\ x_s = 1; \\
&\quad x_s \geq \min_{\lambda_s^a \in \mathcal{D}_s^a} g(\lambda_s^a, \mathbf{x})
\end{aligned} \quad \Rightarrow \quad
\begin{aligned}
&\min_{\mathbf{x}, \lambda} \mathbf{x^T 1} && \\
&\text{s.t. } x_s = 0;\ x_s = 1; && \forall s \in S^{no}; \forall s \in S^{yes}; && \text{(13a)} \\
&\quad x_s \geq g(\lambda_s^a, \mathbf{x}); && \forall s \in S^?, \forall a \in \mathcal{A}(s); && \text{(13b)} \\
&\quad \lambda_s^a \in \mathcal{D}_s^a && \forall s \in S^?, \forall a \in \mathcal{A}(s) && \text{(13c)}
\end{aligned}$$

The decision variables of Problem (13) are both $\mathbf{x}$ and λ_s^a, so the CP formulation is convex only if the dual function $g(\lambda_s^a, \mathbf{x})$ is jointly convex in λ_s^a and $\mathbf{x}$. While this condition cannot be guaranteed for arbitrary uncertainty models, we prove constructively that it holds for the ones considered in the thesis. For example, for the interval model, Problem (13) reads:

$$
\begin{aligned}
&\min_{\mathbf{x},\lambda_\mathbf{s}^\mathbf{a}} \mathbf{x}^\mathbf{T}\mathbf{1} \\
&\text{s.t. } x_s = 0;\ x_s = 1; && \forall s \in S^{no}; \forall s \in S^{yes}; \\
&\quad x_s \geq \lambda_{1,s}^a - (\underline{\mathbf{f}}_a^s)^T \lambda_{\mathbf{2,s}}^\mathbf{a} + (\overline{\mathbf{f}}_a^s)^T \lambda_{\mathbf{3,s}}^\mathbf{a}; && \forall s \in S^?, \forall a \in \mathcal{A}(s); \\
&\quad \mathbf{x} + \lambda_{\mathbf{2,s}}^\mathbf{a} - \lambda_{\mathbf{3,s}}^\mathbf{a} - \lambda_{1,s}^a \mathbf{1} = \mathbf{0}; && \forall s \in S^?, \forall a \in \mathcal{A}(s); \\
&\quad \lambda_{\mathbf{2,s}}^\mathbf{a} \geq \mathbf{0},\ \lambda_{\mathbf{3,s}}^\mathbf{a} \geq \mathbf{0} && \forall s \in S^?, \forall a \in \mathcal{A}(s)
\end{aligned}
$$

which is an LP, so trivially jointly convex in $\mathbf{x}$ and λ_s^a. Analogously, Problem (13) for the ellipsoidal model is a Second-Order Cone Program (SOCP), so again jointly convex in $\mathbf{x}$ and λ_s^a. For the likelihood model, Problem (13) reads:

$$
\begin{aligned}
&\min_{x_s,\lambda_s^a} \mathbf{x}_\mathbf{s}^\mathbf{T}\mathbf{1} \\
&\text{s.t. } x_s = 0;\ x_s = 1; && \forall s \in S^{no}; \forall s \in S^{yes}; && \text{(14a)} \\
&\quad x_s \geq \lambda_{1,s}^a - (1+\beta_s^a)\lambda_{2,s}^a + \lambda_{2,s}^a \textstyle\sum_{s'} h_{ss'}^a \log\left(\frac{\lambda_{2,s}^a h_{ss'}^a}{\lambda_{1,s}^a - x_{s'}}\right); && \forall s \in S^?, \forall a \in \mathcal{A}(s); && \text{(14b)} \\
&\quad \lambda_{1,s}^a \geq \max_{s' \in S} x_{s'};\ \lambda_{2,s}^a \geq 0 && \forall s \in S^?, \forall a \in \mathcal{A}(s) && \text{(14c)}
\end{aligned}
$$

We prove its joint convexity in $\mathbf{x}$ and λ_s^a as follows. The cost function and Constraints (14a)-(14c) are trivially convex. Constraint (14b) is generated by a primal-dual transformation, so, according to convex theory, it is convex in the dual variables λ_s^a by construction. Moreover, convex theory also guarantees that the affine subtraction of $\mathbf{x}$ from λ_s^a preserves convexity, so we conclude that Problem (14) is convex.

For general CMDPs, we will assume:

Assumption 5.1. *Given a CMDP $\mathcal{M}_\mathcal{C}$, for all convex uncertainty sets $\mathcal{F}_s^a \in \mathcal{F}$, the dual function $g(\lambda_s^a, \mathbf{x})$ in Problem (12) is jointly convex in both $\lambda_\mathbf{s}^\mathbf{a}$ and $\mathbf{x}$.*

According to Proposition 4.1, Problem (13) can thus be solved in polynomial time. Also for this formulation, $\mathbf{P}^{max}[\phi_1 \mathcal{U} \phi_2] = \mathbf{x}^*$, so all the satisfaction probabilities can be computed by solving only one convex problem. Finally, we note that we can combine models of uncertainty different from one another within a single CP formulation, since each dual problem is independent from the others according to Assumption 2.1. As an example, if both the interval and ellipsoidal models are used, the overall CP formulation is an SOCP.

Lemma 5.2. *The routine to verify the* Unbounded Until *operator is sound, complete and guaranteed to terminate with algorithmic complexity polynomial in the size $\mathcal{R}$ of $\mathcal{M}_{\mathcal{C}}$, if $\mathcal{M}_{\mathcal{C}}$ satisfies Assumption 5.1.*

Proof. The routine solves only one convex program, which is generated in time polynomial in $\mathcal{R}$ as follows. We formulate Constraints (13b) and (13c) for all $s \in S^?$ and $a \in \mathcal{A}(s)$, i.e., $O(MQ)$ constraints, where $Q = |S^?| = O(N)$. They are derived from MQ primal-dual transformations as in Equation (12). Each primal inner problem has N unknowns, $N+1$ constraints to represent the probability simplex and D_s^a constraints to represent the uncertainty set $\mathcal{F}_s^a$. From duality theory, the corresponding dual inner problem has $N+1+D_s^a$ unknowns and $2N+1+D_s^a$ constraints ($N+1+D_s^a$ constraints are "structural", i.e., in the form $\lambda \leq 0$ or $\lambda \geq 0$ [16]). Overall, Problem (13) has $O\left((N+1+D)MQ\right)$ more unknowns and $O\left((2N+1+D)MQ\right)$ more constraints of Problem (10), so its size is polynomial in $\mathcal{R}$. If $\mathcal{M}_{\mathcal{C}}$ satisfies Assumption 5.1, Problem (13) is convex. Using Lemma 4.1, we then conclude that it can be solved in time polynomial in $\mathcal{R}$. Finally, when strong duality holds for the transformation in Equation (12), soundness and completeness of the final solution are preserved because the dual and primal optimal value of each inner problem are equivalent. □

We verify $\phi = P_{\geq 0.3}[\ \vartheta\ \mathcal{U}\ \omega\ ]$ on the example in Figure 1. Problem (13) written with the data of the model has 19 variables and 11 constraints (attached in Appendix A). The solution reads: $\mathbf{P}^{min}[\ \vartheta\ \mathcal{U}\ \omega\ ] = [0.2, 0, 1, 0.32]$, and, in conclusion, $Sat(\phi) = \{s_2, s_3\}$.

5.3 *Bounded Until* Operator

We present the routine to verify property $\phi = P_{\triangleleft p}[\phi_1 \mathcal{U}^{\leq k} \phi_2]$ on a CMDP of size $\mathcal{R}$. First, the set $S^{yes} \overset{def}{=} Sat(\phi_2)$, $S^{no} \overset{def}{=} S \setminus (Sat(\phi_1) \cup Sat(\phi_2))$ and $S^? = S \setminus (S^{no} \cup S^{yes})$ are precomputed. Second, the maximum probabilities $\mathbf{P}^{max}[\psi] = \mathbf{x}^k = G^k(\mathbf{0})$ to satisfy ϕ are computed for all states $s \in S$ applying k times mapping

G defined as:

$$\mathbf{x}^i = G^i(\mathbf{x}^{i-1}) = \begin{cases} 0;\ 1; & \forall s \in S^{no};\ \forall s \in S^{yes}; \\ 0; & \forall s \in S^? \wedge i = 0; \\ \max\limits_{a\in\mathcal{A}(s)} \max\limits_{\mathbf{f}_s^a \in \mathcal{F}_s^a} (\mathbf{x}^{i-1})^T \mathbf{f_s^a} & \forall s \in S^? \wedge i > 0 \end{cases} \tag{15}$$

and $\mathbf{x}^{-1} = \mathbf{0} \in \mathbb{R}^N$. Finally, the computed probabilities are compared to the threshold p.

Lemma 5.3. *The routine to verify the* Bounded Until *operator is sound, complete and guaranteed to terminate with algorithmic complexity that is polynomial in the size $\mathcal{R}$ of $\mathcal{M}_\mathcal{C}$ and pseudo-polynomial in the time bound k of $\mathcal{U}^{\leq k}$.*

Proof. For the first part, the proof is similar to the one for the *Next* Operator. To apply Mapping (15), we need to solve $O(MQ)$ "inner" convex programs with $Q = \mid S^? \mid$, i.e., one $\forall s \in S^?, a \in \mathcal{A}(s)$. Each problem has at most N unknowns, $N + 1 + D$ constraints, and it is solved in time polynomial in $\mathcal{R}$. Further, the pseudo-polynomial complexity in k comes from applying Mapping (15) k times. While each inner problem can be solved with accuracy $\pm\epsilon_{inn}$ in time linear in $\log(1/\epsilon_{inn})$ by Proposition 4.1, we are left to prove the soundness and completeness of the overall solution, since the ϵ_{inn}-approximations in computing $\mathbf{x}^i, \forall i$ get propagated at each iteration, and might in principle result in a larger error at the end of the procedure. We call ϵ_s^i the error accumulated at step i for state s, $x_s^i = x_{s,id}^i + \epsilon_s^i$, where $x_{s,id}^i$ is the solution with no error propagation, and ϵ_s^k the error in the final solution. Also, $\mathbf{f}_s^{a,i} \in \mathcal{F}_s^a$ is the optimal solution of the inner problem at step i. We solve this difficulty by noting that the optimal value of the the inner problem is computed by multiplying vector $\mathbf{x}^i$ by $\mathbf{f}_s^{a,i} \in \mathcal{F}_s^a$, with $\mathbf{1}^T\mathbf{f}_s^a = 1, \forall \mathbf{f}_s^a \in \mathcal{F}_s^a, \forall a \in \mathcal{A}(s)$. At the first, second and i^{th} iteration:

$$\begin{aligned} &x_s^1 = x_{s,id}^1 + \epsilon_s^1 = \mathbf{f}_s^{a,1}\mathbf{x}^0 + \epsilon_{inn} \\ &x_s^2 = \mathbf{f}_s^{a,2}\mathbf{x}^1 + \epsilon_{inn} = \mathbf{f}_s^{a,2}\left(\mathbf{f}_s^{a,1}\mathbf{x}^0 + \epsilon_{inn}\mathbf{1}\right) + \epsilon_{inn} = \mathbf{f}_s^{a,2}\mathbf{f}_s^{a,1}\mathbf{x}^0 + 2\epsilon_{inn} \\ &x_s^i = \mathbf{f}_s^{a,i}\mathbf{x}^{i-1} + \epsilon_{inn} = \mathbf{f}_s^{a,i}\left(\mathbf{f}_s^{a,i-1}\mathbf{x}^{i-1} + (i-1)\epsilon_{inn}\mathbf{1}\right) + \epsilon_{inn} = \mathbf{f}_s^{a,i}\mathbf{f}_s^{a,i-1}\ldots\mathbf{f}_s^{a,1}\mathbf{x}^0 + i\epsilon_{inn} \end{aligned}$$

so ϵ_s^i increases linearly with i. The desired accuracy ϵ_d of the final solution can thus be guaranteed by solving each inner problem with accuracy $\epsilon_{inn} = \epsilon_d/k$. □

We verify $\phi = P_{\leq 0.6}[\vartheta\ \mathcal{U}^{\leq 1}\omega]$ in the example in Fig. 1. We get $S^{yes} = \{s_2\}$, $S^{no} = \{s_1\}$. Applying once Mapping (15), we get $\mathbf{P}^{max}[\vartheta\mathcal{U}^{\leq 1}\omega] = [0.4, 0, 1, 0.6]$ and $Sat(\phi) = \{s_0, s_1, s_3\}$.

Chapter 6
Case Studies

In this chapter, we present an experimental evaluation of the functionality of the proposed algorithms. We first describe the experimental setup. We then present two case studies. In the first case study, we analyze the performance of a distributed stochastic consensus protocol where one of the participating processes is faulty or behaving maliciously. In the second case study, we analyze the performance of a wireless protocol operating across a lossy physical link.

We implemented the proposed verification algorithm in Python, and interfaced it with PRISM [6] to extract information about the CMDP model. We used MOSEK [34] to solve the LPs generated for the interval model and implemented customized numerical solvers for the other models of uncertainty. The implemented tool is available at [35]. The algorithm was tested on all the case studies collected in the PRISM benchmark suite [36]. In this thesis, we report two of them: the verification of a consensus protocol and of a dynamic configuration protocol for IPv4 addresses. The goals of these experiments are two-fold:

1. quantitatively evaluate the impact of uncertainty on the results of verification of PCTL properties of CMDPs;
2. assess the scalability of the proposed approach to increasing problem size.

The runtime data were obtained on a 2.4 GHz Intel Xeon with 32GB of RAM.

6.1 Consensus Protocol

Consensus problems arise in many distributed environments, where a group of distributed processes attempt to reach an agreement about a decision to take by accessing some shared entity. A consensus protocol ensures that the processes will eventually terminate and take the same decision, even if they start with initial guesses that might differ from one another.

We analyze the randomized consensus protocol presented in [22, 37]. The protocol guarantees that the processes return a preference value $v \in \{1, 2\}$, with probability parameterized by a process independent value R $(R \geq 2)$ and the number of processes P. The processes communicate with one another by accessing a shared counter of value c. The protocol proceeds in rounds. At each round, a process flips a local coin, increments or decrements the shared counter depending on the outcome and then reads its value c. If $c \geq PR$ ($c \leq -PR$), it chooses $v = 1$ ($v = 2$). Note that the larger the value of R, the longer it takes on average for the processes to reach the decision. Nondeterminism is used to model the asynchronous access of the processes to the shared counter, so the overall protocol is modeled as an MDP.

We verify the property **Agreement:** all processes must agree on the same decision, i.e., choose a value $v \in \{1, 2\}$. We compute the minimum probability of **Agreement** and compare it against the theoretical lower bound $(R - 1)/2R$ [22]. In PCTL syntax:

$$P_{s_0}^{min}[\psi] := P_{s_0}^{min}(\mathbf{F}\ (\{finished\} \wedge \{all\_coins\_equal\_1\})) \tag{16}$$

We consider the case where one of the processes is unreliable or adversarial, i.e., it throws a biased coin instead of a fair coin. Specifically, the probability of either outcome lies in the uncertainty interval $[(1 - u)p_0, (1 + u)p_0]$, where $p_0 = 0.5$ according to the protocol. This setting is relevant to analyze the protocol robustness when a process acts erroneously due to a failure or a security breach. In particular, our approach allows to study attacks that deliberately hide under the noise threshold of the protocol. In such attacks, the compromised node defers agreement by producing outputs whose statistical properties are within the noise tolerance of an uncompromised node, so that it is harder to detect its malicious behavior.

Figure 2 shows the effect of different levels of uncertainty on the computed probabilities for $P = 4$. With no uncertainty ($u = 0$), $P_{s_0}^{min}$ increases as R increases, because a larger R drives the decision regions further apart, making it more difficult for the processes to decide on different values of v. As R goes to infinity, $P_{s_0}^{min}$ approaches the theoretical lower bound $lim_{R\to\infty}(R-1)/2R = 0.5$. However, even with a small uncertainty ($u = 0.01$), $P_{s_0}^{min}$ soon decreases for increasing R. With a large uncertainty ($u = 0.15$), $P_{s_0}^{min}$ quickly goes to 0. A possible explanation is that the faulty process has more opportunities to deter agreement for a high R, since R also determines the expected time to termination. Results thus show that the protocol is vulnerable to uncertainties. This fact may have serious security implication, i.e., a denial-of-service attack could reduce the availability of the distributed service, since a compromised process may substantially alter the expected probability of agreement.

Lastly, we study the scalability of the CP procedure, by evaluating Equation (16) while sweeping R both for $P = 2$ and $P = 4$. We use MOSEK [34] to solve Problem (13) and set the Time Out (TO) to one hour. In Figure 3, we plot the sum

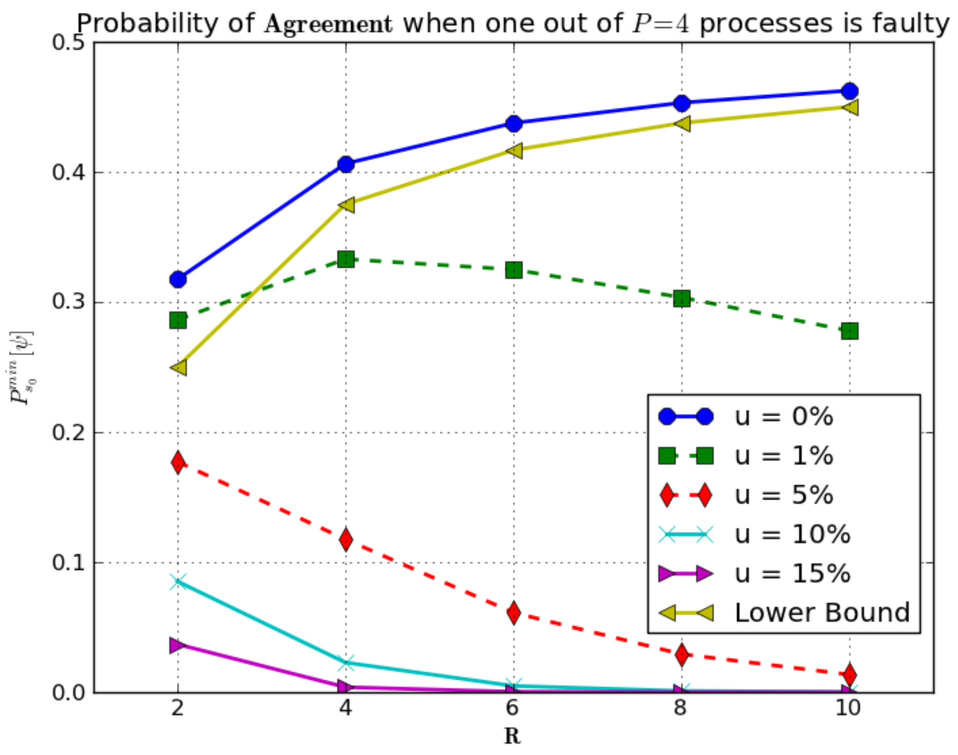

Figure 2: Value of Equation (16) as a function of the value of parameter R while varying the uncertainty level u.

$(N + T)$ of the number of states (N) and transitions (T) of the CMDP, which are independent of the uncertainty in the transition probabilities, to represent the model size (top), the sum $(V+C)$ of the number of variables (V) and constraints (C) of the generated LP instances of Problem (13) (center), and the running time t_{CP} (bottom). $V+C$ always scales linearly with $N+T$ (the lines have the same slope), supporting the polynomial complexity result for our algorithm. Instead, t_{CP} scales linearly only for smaller problems $(P = 2)$, while it has a higher-order polynomial behavior for larger problems $(P = 4)$ (the line is still a straight line but with steeper slope, so it is polynomial on logarithmic axes). This behavior depends on the performance of the chosen numerical solver, and it can improve benefiting of future advancements in the solver implementation. In Table 3, we compare the CP procedure with two tools, PRISM [6] and PARAM [38], in terms of runtime, for varying values of P and R. Although neither tool solves the same problem addressed in this thesis, the comparison is useful to assess the practicality of the proposed approach. In

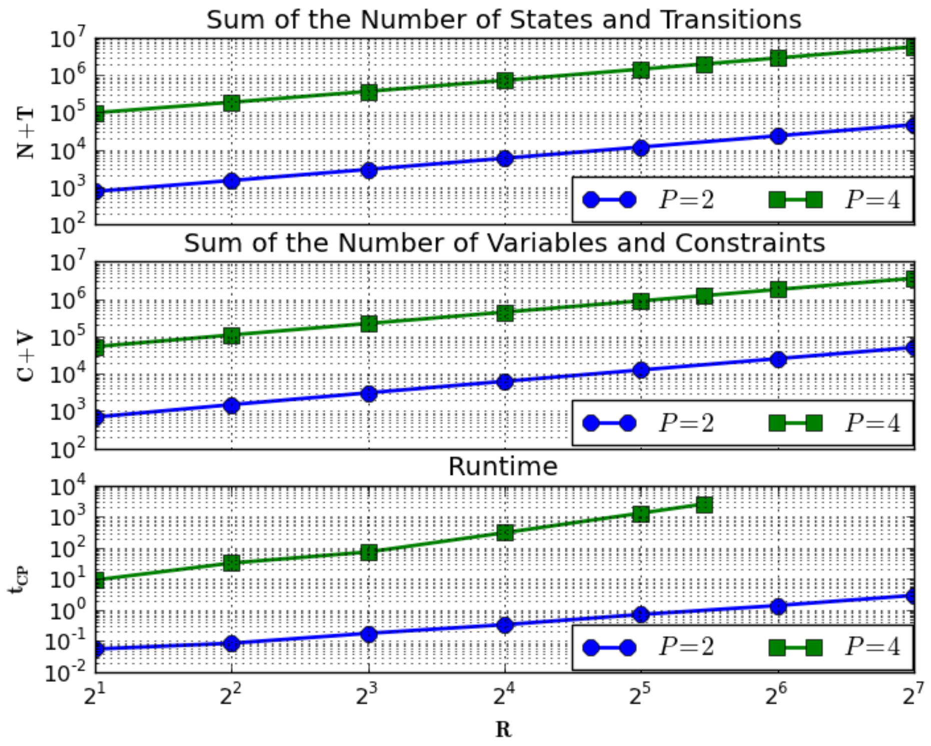

Figure 3: Analysis of the scalability of the CP procedure.

particular, PRISM only verifies PCTL properties of MDPs with no uncertainties. PARAM instead derives a symbolic expression of the satisfaction probabilities as a function of the model parameters, to then find the parameter values that satisfy the property. Hence, PRISM only considers a special case of the models considered in this work, while our approach only returns the worst-case scenario computed by PARAM. Results show that the CP procedure runs faster than PRISM for some benchmarks, but it is slower for larger models. This is expected since the scalability of our approach depends mainly on the problem size, while the performance of the iterative engine in PRISM depends on the problem size and on the number of iterations required to achieve convergence, which is dependent on the problem data. Finally, our approach is orders of magnitude faster than PARAM, so it should be preferred to perform worst-case analysis of system performances.

Table 3: Runtime Comparison

Tool	$P=2, R=2$ $N+T=764$	$R=7$ $2,604$	$R=128$ $47,132$	$P=4, R=2$ $97,888$	$R=32$ $1,262,688$	$R=44$ $1,979,488$	$P=6, R=4$ $14,211,904$
CP	$0.02s$	$0.1s$	$2.1s$	$8.3s$	$1,341s$	$2,689$	TO
PRISM	$0.01s$	$0.09s$	$196s$	$1s$	$2,047s$	TO	$1860s$
PARAM	$22.8s$	$657s$	TO	TO	TO	TO	TO

6.2 ZeroConf Dynamic Configuration Protocol for IPv4 Link-Local Addresses

The ZeroConf protocol [39, 40] is an Internet Protocol (IP)-based configuration protocol for local (e.g. domestic) networks. In such a local context, each device should configure its own unique IP address when it gets connected to the network, with no user intervention. The protocol thus offers a distributed "plug-and-play" solution in which address configuration is managed by individual devices when they are connected to the network. The network is composed of DV_{tot} devices. After being connected, a new device chooses randomly an IP address from a pool of $IP_A = 65024$ available ones, as specified by the standard. The address is non-utilized with probability $p_0 = 1 - DV_{tot}/IP_A$. It then sends messages to the other devices in the network, asking whether the chosen IP address is already in use. If no reply is received, the device starts using the IP address, otherwise the process is repeated.

The protocol is both probabilistic and timed: probability is used in the randomized selection of an IP address and to model the eventuality of message loss; timing defines intervals that elapse between message retransmissions. In [40], the protocol has been modeled as an MDP using the digital clock semantic of time. In this semantic, time is discretized in a finite set of epochs which are mapped to a finite number of states in an MDP, indexed by the epoch variable t_e. To enhance the user experience and, in battery-powered devices, to save energy, it is important to guarantee that a newly-connected device manages to select a unique IP address within a given deadline *dl*. For numerical reasons, we study the maximum probability of *not* being able to select a valid address within *dl*. In PCTL syntax:

$$P^{max}_{s_0}[\psi] := P^{max}_{s_0}(\neg\{unique\_address\}\,\mathcal{U}\,\{t_e > dl\}) \tag{17}$$

We analyzed how network performances vary when there is uncertainty in estimating: 1) the probability of selecting an IP address, and; 2) the probability of message loss during transmission. The former may be biased in a faulty or malicious device. The latter is estimated from empirical data, so it is approximated. Further, the

IMDP semantic of IDTMCs (Section 1), which allows a nature to select a different transition distribution at each execution step, properly models the time-varying characteristics of the transmission channel.

In Figure 4, we added uncertainty only to the probability of message loss using the likelihood model, which is suitable for empirically-estimated probabilities. Using classical results from statistics [7], we computed the value of parameter β from Set (2) corresponding to several confidence levels C_L in the measurements. In particular, $0 \leq C_L \leq 1$ and $C_L = 1 - cdf_{\chi^2_d}(2 * (\beta_{max} - \beta))$, where $cdf_{\chi^2_d}$ is the cumulative density function of the Chi-squared distribution with d degrees of freedom ($d = 2$ here because there are two possible outcomes, message lost or received). Results show that the value of $P^{max}_{s_0}$ increases by up to $\sim 10\times$ for decreasing C_L, while classical model-checking would only report the value for $C_L = 1$, which coarsely over-estimates network performance. The plot can be used by a designer to choose *dl* to make the protocol robust to varying channel conditions, or by a field engi-

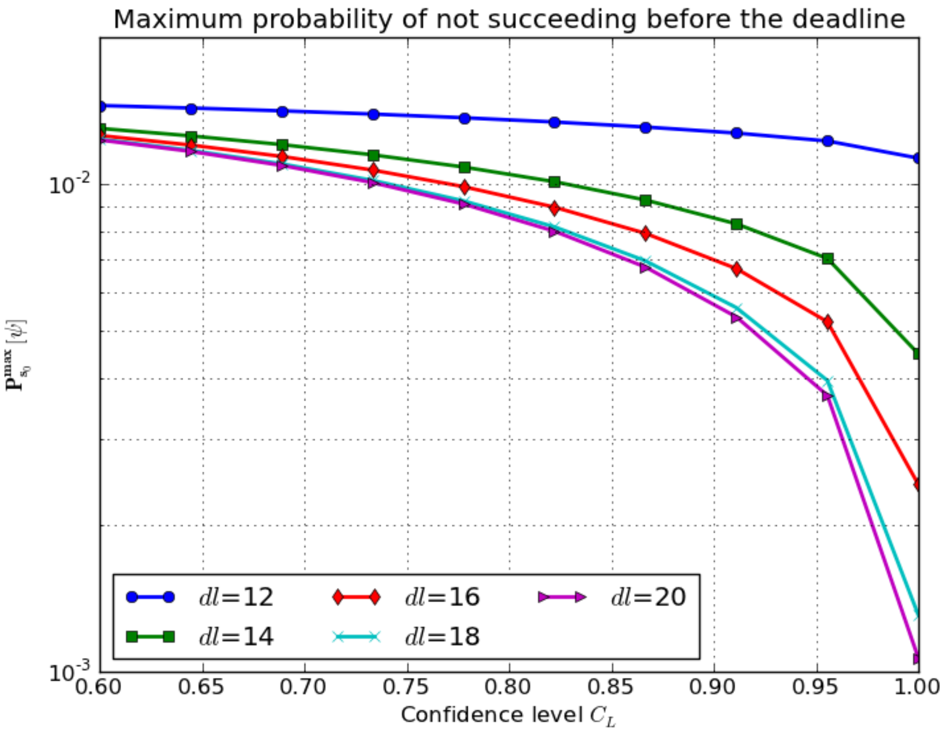

Figure 4: Value of Equation (17) as a function of the confidence level C_L.

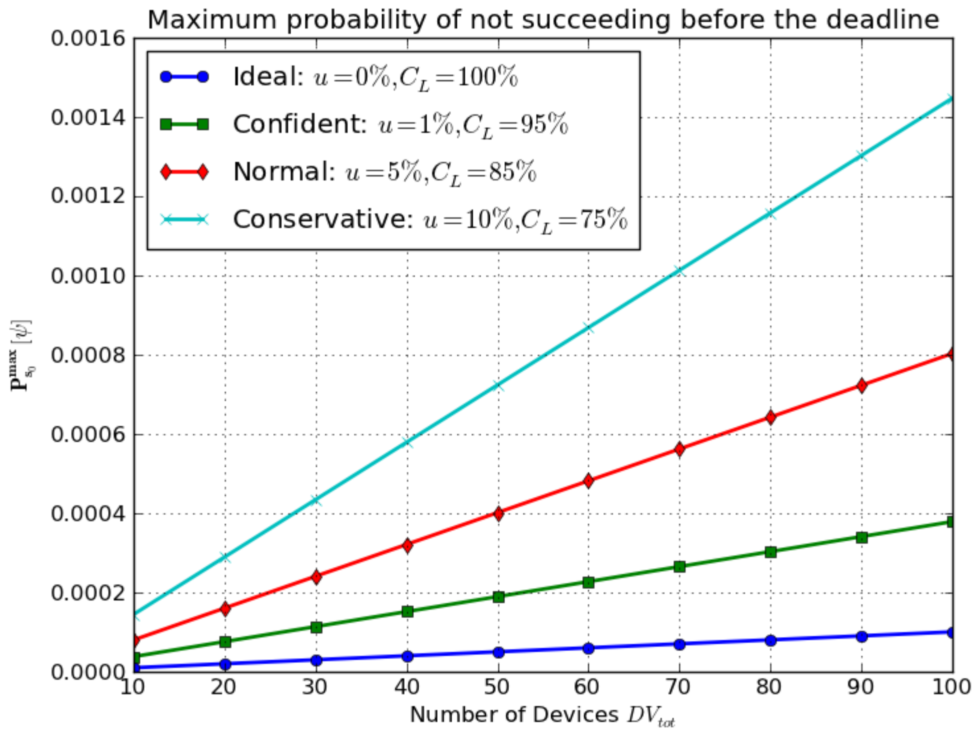

Figure 5: Value of Equation (17) for increasing number of devices in the network.

neer to assess when the collected measurements are enough to estimate network performances.

In Figure 5, we compose different models of uncertainty, i.e., we also add uncertainty in the probability of selecting the new IP address using the interval model. This probability thus lies in the interval $[(1-u)p_0,(1+u)p_0]$. We, arbitrarily, fixed $dl = 25$ and swept DV_{tot} in the range $[10-100]$, which covers most domestic applications, to study how network congestion affects the value of Equation (17). We studied four scenarios: the *ideal* scenario, returned by classical model-checking techniques; the *confident, normal, conservative* scenarios, where we added increasing uncertainty to model different knowledge levels of the network behavior, a situation that often arises during the different design phases, from conception to deployment. Results show that $P_{s_0}^{max}[\psi]$ gets up to $\sim\ 15\times$ higher than the ideal scenario, an information that designers can use to determine the most sensitive parameters of the system and to assess the impact of their modeling assumptions on the estimation of network performances.

Chapter 7
Conclusions and Future Work

We addressed the problem of verifying PCTL properties of Markov Decision Processes whose state transition probabilities are only known to lie within uncertainty sets. We considered the class of Convex-MDPs (CMDPs), i.e., MDPs with convex uncertainties. Using results on strong duality for convex programs, we proved that model checking is decidable in PTIME for the fragment of PCTL without the *Bounded Until* operator. For the entire PCTL syntax, the algorithmic complexity becomes pseudo-polynomial in the size of the property. We validated our approach on two case studies. Results show that uncertainty substantially alters the computed probabilities, thus revealing the importance of the proposed analysis.

As future work, we aim to relax the *rectangular uncertainty* assumption, to limit the adversarial choices of state transition probability distributions, i.e, the action range of nature, and obtain a less conservative and more realistic analysis. Also, we plan to verify a complex physical system, e.g. an airplane power system, in which modeling uncertainties are present both in the underlying physical process and in the failure probabilities of its components.

References

1. A. Puggelli, W. Li, A. Sangiovanni-Vincentelli, and S. Seshia, "Polynomial-time verification of pctl properties of mdps with convex uncertainties," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Springer Berlin Heidelberg, 2013, vol. 8044, pp. 527–542. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39799-8_35
2. C. Courcoubetis and M. Yannakakis, "The Complexity of Probabilistic Verification," *J. ACM*, vol. 42, no. 4, pp. 857–907, Jul. 1995. [Online]. Available: http://doi.acm.org/10.1145/210332.210339
3. A. Bianco and L. Alfaro, "Model Checking of Probabilistic and Nondeterministic Systems," in *Foundations of Software Technology and Theoretical Computer Science*, ser. Lecture Notes in Computer Science, P. Thiagarajan, Ed. Springer Berlin Heidelberg, 1995, vol. 1026, pp. 499–513. [Online]. Available: http://dx.doi.org/10.1007/3-540-60692-0_70
4. M. Kwiatkowska, "Quantitative Verification: Models, Techniques and Tools," in *Proc. 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM Press, September 2007, pp. 449–458.
5. H. Hansson and B. Jonsson, "A Logic for Reasoning About Time and Reliability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994. [Online]. Available: http://dx.doi.org/10.1007/BF01211866
6. M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-Time Systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.
7. A. Nilim and L. El Ghaoui, "Robust Control of Markov Decision Processes with Uncertain Transition Matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005. [Online]. Available: http://pubsonline.informs.org/doi/abs/10.1287/opre.1050.0216
8. I. Kozine and L. Utkin, "Interval-Valued Finite Markov Chains," *Reliable Computing*, vol. 8, no. 2, pp. 97–113, 2002. [Online]. Available: http://dx.doi.org/10.1023/A%3A1014745904458
9. K. Sen, M. Viswanathan, and G. Agha, "Model-Checking Markov Chains in the Presence of Uncertainties," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, H. Hermanns and J. Palsberg, Eds. Springer Berlin Heidelberg, 2006, vol. 3920, pp. 394–410. [Online]. Available: http://dx.doi.org/10.1007/11691372_26

10. K. Chatterjee, K. Sen, and T. Henzinger, “Model-Checking ω-Regular Properties of Interval Markov Chains,” in *Foundations of Software Science and Computational Structures*, ser. Lecture Notes in Computer Science, R. Amadio, Ed. Springer Berlin Heidelberg, 2008, vol. 4962, pp. 302–317. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78499-9_22
11. A. Ben-Tal and A. Nemirovski, “Robust Solutions of Uncertain Linear Programs,” *Operations Research Letters*, vol. 25, no. 1, pp. 1 – 13, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167637799000164
12. V. Kreinovich, A. Neumaier, and G. Xiang, “Towards a Combination of Interval and Ellipsoid Uncertainty,” Department of Computer Science, UT-El Paso, Tech. Rep. UTEP-CS-07-42b, 2008.
13. A. Andreychenko, L. Mikeev, D. Spieler, and V. Wolf, “Parameter Identification for Markov Models of Biochemical Reactions,” in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, G. Gopalakrishnan and S. Qadeer, Eds. Springer Berlin Heidelberg, 2011, vol. 6806, pp. 83–98. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22110-1_8
14. A. McCallum, D. Freitag, and F. C. N. Pereira, “Maximum Entropy Markov Models for Information Extraction and Segmentation,” in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 591–598. [Online]. Available: http://dl.acm.org/citation.cfm?id=645529.658277
15. J. Gill, “An Entropy Measure of Uncertainty in Vote Choice,” *Electoral Studies*, vol. 24, pp. 371–392, 2005.
16. S. Boyd and L. Vandenberghe, “Convex Optimization,” *Cambridge University Press*, 2004.
17. M. Vardi, “Automatic Verification of Probabilistic Concurrent Finite State Programs,” in *Foundations of Computer Science, 1985., 26th Annual Symposium on*, Oct 1985, pp. 327–338.
18. E. Lehmann and G. Casella, *Theory of Point Estimation*. Springer-Verlag, New York, 1998.
19. M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.
20. V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker, “Automated Verification Techniques for Probabilistic Systems,” in *Formal Methods for Eternal Networked Software Systems (SFM’11)*, ser. LNCS, M. Bernardo and V. Issarny, Eds., vol. 6659. Springer, 2011, pp. 53–113.
21. E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.
22. M. Kwiatkowska, G. Norman, and R. Segala, “Automated Verification of a Randomized Distributed Consensus Protocol Using Cadence SMV

and PRISM," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, G. Berry, H. Comon, and A. Finkel, Eds. Springer Berlin Heidelberg, 2001, vol. 2102, pp. 194–206. [Online]. Available: http://dx.doi.org/10.1007/3-540-44585-4_17

23. M. Lahijanian, S. Andersson, and C. Belta, "Temporal Logic Motion Planning and Control With Probabilistic Satisfaction Guarantees," *Robotics, IEEE Transactions on*, vol. 28, no. 2, pp. 396–409, April 2012.
24. E. Hahn, H. Hermanns, B. Wachter, and L. Zhang, "INFAMY: An Infinite-State Markov Model Checker," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, A. Bouajjani and O. Maler, Eds. Springer Berlin Heidelberg, 2009, vol. 5643, pp. 641–647. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02658-4_49
25. J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen, "The Ins and Outs of the Probabilistic Model Checker MRMC," *Perform. Eval.*, vol. 68, no. 2, pp. 90–104, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2010.04.001
26. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle, "ETMCC: Model Checking Performability Properties of Markov Chains." in *DSN*. IEEE Computer Society, 2003, pp. 673–. [Online]. Available: http://dblp.uni-trier.de/db/conf/dsn/dsn2003.html#HermannsKMS03
27. K. Sen, M. Viswanathan, and G. Agha, "VESTA: A Statistical Model-Checker and Analyzer for Probabilistic Systems," in *Quantitative Evaluation of Systems, 2005. Second International Conference on the*, Sept 2005, pp. 251–252.
28. H. Younes, "Ymer: A Statistical Model Checker," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, K. Etessami and S. Rajamani, Eds. Springer Berlin Heidelberg, 2005, vol. 3576, pp. 429–433. [Online]. Available: http://dx.doi.org/10.1007/11513988_43
29. E. Wolff, U. Topcu, and R. Murray, "Robust Control of Uncertain Markov Decision Processes with Temporal Logic Specifications," *Intl. Conf. on Decision and Control (CDC)*, 2012.
30. A. D'Innocenzo, A. Abate, and J.-P. Katoen, "Robust PCTL Model Checking," in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '12. New York, NY, USA: ACM, 2012, pp. 275–286. [Online]. Available: http://doi.acm.org/10.1145/2185632.2185673
31. E. Clarke and E. Emerson, "Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic," in *Logics of Programs*, ser. Lecture Notes in Computer Science, D. Kozen, Ed. Springer Berlin Heidelberg, 1982, vol. 131, pp. 52–71. [Online]. Available: http://dx.doi.org/10.1007/BFb0025774

32. Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611970791
33. A. Puggelli, W. Li, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, “Polynomial-Time Verification of PCTL Properties of MDPs with Convex Uncertainties,” UC-Berkeley, Tech. Rep. UCB/EECS-2013-24, Apr 2013. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-24.html
34. “MOSEK,” http://www.mosek.com.
35. Online: http://www.eecs.berkeley.edu/~puggelli/.
36. Online: http://www.prismmodelchecker.org/benchmarks/.
37. J. Aspnes and M. Herlihy, “Fast Randomized Consensus Using Shared Memory,” *Journal of Algorithms*, vol. 11, no. 3, pp. 441 – 461, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0196677490900216
38. E. Hahn, T. Han, and L. Zhang, “Synthesis for pctl in parametric markov decision processes,” in *NASA Formal Methods*, ser. Lecture Notes in Computer Science, M. Bobaru, K. Havelund, G. Holzmann, and R. Joshi, Eds. Springer Berlin Heidelberg, 2011, vol. 6617, pp. 146–161. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20398-5_12
39. S. Cheshire, B. Adoba, and E. Gutterman, “Dynamic Configuration of IPv4 Link Local Addresses,” available from http://www.ietf.org/rfc/rfc3927.txt.
40. M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, “Performance Analysis of Probabilistic Timed Automata Using Digital Clocks,” *Formal Methods in System Design*, vol. 29, pp. 33–78, 2006.

Appendices

A Example of a Full Linear-Program Formulation

This appendix reports the full linear-program formulation that was used in Section 5.2 to verify property $\phi = P_{\geq 0.3}[\ \vartheta\ \mathcal{U}\ \omega\ ]$ on the example in Figure 1. Problem (13) written with the data of the model has 19 variables and 11 constraints. All variables are (implicitly) bounded to be positive apart from the ones labeled as *free* at the bottom of the formulation.

$$
\begin{aligned}
&\max_{\mathbf{x},\lambda_1,\boldsymbol{\lambda_2},\boldsymbol{\lambda_3}} \ x_0 + x_3 && (18)\\
&\text{Subject to:}\\
&\quad x_2 = 1\\
&\quad x_1 = 0\\
&\quad x_0 \leq \lambda^a_{1,s_0} + 0.6\lambda^a_{2,s_0s_0} + 0.2\lambda^a_{2,s_0s_1} - 0.8\lambda^a_{3,s_0s_0} - 0.5\lambda^a_{3,s_0s_1}\\
&\quad x_1 - \lambda^a_{1,s_0} + \lambda^a_{3,s_0s_0} - \lambda^a_{2,s_0s_0} = 0\\
&\quad x_2 - \lambda^a_{1,s_0} + \lambda^a_{3,s_0s_1} - \lambda^a_{2,s_0s_1} = 0\\
&\quad x_0 \leq +x_3\\
&\quad x_3 \leq \lambda^a_{1,s_3} + 0.1\lambda^a_{2,s_3s_0} + 0.5\lambda^a_{2,s_3s_1} + 0.3\lambda^a_{2,s_3s_2} - 0.5\lambda^a_{3,s_3s_0} - 0.8\lambda^a_{3,s_3s_1} - 0.4\lambda^a_{3,s_3s_2}\\
&\quad x_0 - \lambda^a_{1,s_3} + \lambda^a_{3,s_3s_0} - \lambda^a_{2,s_3s_0} = 0\\
&\quad x_1 - \lambda^a_{1,s_3} + \lambda^a_{3,s_3s_1} - \lambda^a_{2,s_3s_1} = 0\\
&\quad x_2 - \lambda^a_{1,s_3} + \lambda^a_{3,s_3s_2} - \lambda^a_{2,s_3s_2} = 0\\
&\quad x_3 \leq \lambda^b_{1,s_3} + 0.3\lambda^b_{2,s_3s_0} + 0.4\lambda^b_{2,s_3s_1} - 0.7\lambda^b_{3,s_3s_0} - 0.6\lambda^b_{3,s_3s_1}\\
&\quad x_2 - \lambda^b_{1,s_3} + \lambda^b_{3,s_3s_0} - \lambda^b_{2,s_3s_0} = 0\\
&\quad x_3 - \lambda^b_{1,s_3} + \lambda^b_{3,s_3s_1} - \lambda^b_{2,s_3s_1} = 0\\
&\text{Free: } \lambda^a_{1,s_0}, \lambda^a_{1,s_3}, \lambda^b_{1,s_3}
\end{aligned}
$$