Efficient Probabilistic Model Checking of Systems with Ranged Probabilities

Khalil Ghorbal[1], Parasara Sridhar Duggirala[1,2], Vineet Kahlon[1], Franjo Ivančić[1], and Aarti Gupta[1]

[1] NEC Laboratories America, Inc.
[2] University of Illinois at Urbana Champaign

Abstract. We introduce a new technique to model check reachability properties on Interval Discrete-Time Markov Chains (IDTMC). We compute a sound over-approximation of the probabilities of satisfying a given property where the accuracy is characterized in terms of error bounds. We leverage affine arithmetic to propagate the first-order error terms. Higher-order error terms are bounded using interval arithmetic.

1 Introduction

Analyzing the behavior of real world systems, such as energy management systems or cloud-based web applications, is of great importance for both designers and managers of these systems. Many properties of interest such as performance and reliability, are related to the inherent stochastic behavior of these systems. However, many of these complex systems do not have a readily available model that captures these behaviors. Even if such models exist, they are usually deprecated and do not reflect the actual behavior of the system, partly because these systems get updated and tuned very often after the initial deployment.

Usually, the only available information about the system are its runtime logs, which are systematically recorded either for debugging reasons or for helping in their operational management. Techniques like Statistical Model Checking (SMC) [12,20–22] can use these logs to verify whether a system meets a given probabilistic property. Although, SMC is efficient, it does not provide a model of the system, and instead considers it as a black-box.

This paper advocates a model-based approach, where the stochastic behavior of the system is captured using a learned Markov model, specifically an *Interval-Valued Discrete-Time Markov Chain* (IDTMC). IDTMCs were introduced in [14, 16] to allow a realistic encoding of stochastic systems. More recently, IDTMC (called Abstract DTMC) were used for model checking of DTMCs for abstraction purposes to overcome the state space explosion problem [6, 15].

An IDTMC is a classical discrete-time Markov chain with ranged probabilities, where a transition between two states is associated with an interval in which the actual transition probability must lie. In accordance with the *Unified Markov Chains* [14] (UMC) semantics, an IDTMC is considered as a set of possibly infinitely many Discrete-Time Markov Chains (DTMC).

A. Finkel, J. Leroux, and I. Potapov (Eds.): RP 2012, LNCS 7550, pp. 107–120, 2012.

In this work, we focus on model checking learned IDTMCs, where the intervals come mainly from the stochastic uncertainty related to the underlying learning technique. We use approximation-based techniques to compute a sound over-approximation of the probabilities of satisfying a given reachability property, where accuracy is characterized in terms of error bounds. Our technique leverages affine arithmetic, which has been successfully applied in various different domains (such as abstract interpretation [7, 9], numerical validation [4] or reachability analysis of hybrid systems [8]), to precisely propagate uncertainties during computations.

2 Preliminaries

We first define a DTMC as well as an IDTMC.

Definition 1 (DTMC). *A DTMC is a* 4*-tuple:* $\boldsymbol{M} \stackrel{\text{def}}{=} (S, s_0, P, \ell)$*, where* S *is a finite set of states,* $s_0 \in S$ *the initial state,* P *a stochastic matrix, and* $\ell : S \to 2^{AP}$ *is a labelling function which assigns to each state* $s \in S$ *a set of atomic propositions* $a \in AP$ *that are valid in* s*, and* AP *denotes a finite set of atomic propositions. The element* p_{ij} *of the square matrix* P *denotes the transition probability from state* s_i *to state* s_j*. Therefore,* $p_{ij} \in [0, 1]$ *and for all* i*,* $\sum_j p_{ij} = 1$*.*

Definition 2 (IDTMC). *An IDTMC is* 4*-tuple:* $\boldsymbol{M} \stackrel{\text{def}}{=} (S, s_i, \boldsymbol{P}, \ell)$*, where* $\boldsymbol{P}$ *is an interval-valued matrix. It is defined as the following set of DTMCs:*

$$\{M \mid M = (S, s_i, P, \ell), P \in \boldsymbol{P}\} \ .$$

PCTL is a very expressive logic allowing a combination of standard temporal operators and probabilities [11]. For example, one can express whether the probability of a given path formula to be satisfied is greater than (or equal to) a given threshold. In this work, we target non-nested probabilities of reachability properties. Therefore, we only consider the probabilistic properties $P_{\bowtie\gamma}[\psi]$ for

$$\begin{aligned} \phi &::= true \mid a \mid \neg\phi \mid \phi \wedge \phi \\ \psi &::= \mathcal{X}\phi \mid \phi \ \mathcal{U}^{\leq k} \phi \end{aligned}$$

where $a \in AP, \bowtie \in \{<, \leq, >, \geq\}, \gamma \in [0, 1]$ a threshold probability, and $k \in \mathbb{N} \cup \{\infty\}$.

The semantics of the P operator, with respect to a DTMC M, is as follows. The notation $s \models \phi$ means that the state s satisfies the state formula ϕ. A path σ in M is a sequence of (possibly infinitely many) states. The ith state, $i \geq 0$, of σ is denoted by $\sigma[i]$.

$$\begin{aligned} s &\models true && \text{true for all states.} \\ s &\models a && \iff a \in \ell(s) \\ s &\models \neg\phi && \iff s \not\models \phi \\ s &\models \phi_1 \wedge \phi_2 && \iff s \models \phi_1 \wedge s \models \phi_2 \\ \sigma &\models \mathcal{X}\phi && \iff \sigma[1] \models \phi \\ \sigma &\models \phi_1 \ \mathcal{U}^{\leq k} \phi_2 && \iff \exists i, 0 \leq i \leq k : \forall j < i, \\ & && \qquad s_i \models \phi_2 \wedge s_j \models \phi_1 \end{aligned}$$

Let $Prob_M(s, \psi)$ denote the probability that a random path σ in M starting from s ($\sigma[0] = s$) satisfies ψ, i.e. $\sigma \models \psi$.

$$s \models P_{\bowtie\gamma}[\psi] \qquad \Longleftrightarrow Prob_M(s, \psi) \bowtie \gamma$$

Following the UMC semantics [14], an IDTMC is considered as a set of DTMCs. A property is valid with respect to an IDTMC $\boldsymbol{M}$ if it is valid for every DTMC $M \in \boldsymbol{M}$.

$$\boldsymbol{M}, s \models \phi \qquad \Longleftrightarrow \forall M \in \boldsymbol{M} : M, s \models \phi \ .$$

Therefore, model checking a probabilistic property $P_{\bowtie\gamma}(\psi)$ requires computing the set:

$$\{p | p = Prob_M(s, \psi), \forall M \in \boldsymbol{M}\} \ . \tag{1}$$

2.1 Model Checking a DTMC

We recall in this section the standard techniques used to model check a DTMC [11]. We will then discuss the extension to IDTMCs.

For a DTMC M, if $\psi = \mathcal{X}\phi$, then $Prob_M(s_i, \mathcal{X}\phi) = \sum_{s_j \models \phi} p_{ij}$.

If $\psi = \phi_1 \ \mathcal{U}\phi_2$, we first split the set of states as follows.

- $S_{yes} \stackrel{\text{def}}{=} \{s_i \mid s_i \models \phi_2\}$,
- $S_{no} \stackrel{\text{def}}{=} \{s_i \mid s_i \not\models \phi_1 \wedge s_i \not\models \phi_2\}$,
- $S_{maybe} \stackrel{\text{def}}{=} S \setminus (S_{yes} \cup S_{no})$.

If $s_i \in S_{yes}$, then $Prob_M(s_i, \psi) = 1$, and if $s_i \in S_{no}$, then $Prob_M(s_i, \psi) = 0$. The set S_{maybe} denotes the set of states where any path starting from s_i may or may not satisfy the path formula ψ. Therefore, the probability of the satisfiability of the path formula ψ starting from these states is unknown and needs to be computed.

Let $v_k[i]$, the ith component of the vector v_k, denote $Prob_M(s_i, \psi, k)$, that is the probability that a path of length k, starting from a state s_i, satisfies the property ψ. Let i be within $I_{maybe} \stackrel{\text{def}}{=} \{i \mid s_i \in S_{maybe}\}$. Then,

$$v_k[i] = \sum_{j=1}^{n} p_{ij} v_{k-1}[j] \tag{2}$$

$$= \sum_{j \in I_{maybe}} p_{ij} v_{k-1}[j] + \underbrace{\sum_{j \notin I_{maybe}} p_{ij} v_{k-1}[j]}_{b_i} \ . \tag{3}$$

Since $v_{k-1}[j]$ are known for $j \notin I_{maybe}$ (either 0 ot 1), and are independent from k, the quantity b_i is just a constant.

Using a matrix notation for all the states $s_i \in S_{maybe}$, we obtain

$$v_k = P' v_{k-1} + b, \tag{4}$$

where the square matrix P' is simply extracted from the transition probability matrix P by deleting all the rows i, such that $s_i \in S_{yes} \cup S_{no}$, then deleting the columns i such that $s_i \in S_{yes} \cup S_{no}$. The components b_i of the vector b, are defined in equation (3).

Bounded Case. In the bounded case ($k < +\infty$), we unroll the recursion of equation (4) completely starting with $v_0 = 0$. Indeed, the probability that a path of length zero satisfies the property ψ is zero for all states in S_{maybe}. The probability $Prob_M(s_i, \psi, k)$ is then given by the ith component of vector v_k.

Unbounded Case. For the unbounded case, we need to resolve the following system of linear equations

$$v = P'v + b \ .$$

Observe first that for any given state $s_i \in S_{maybe}$, if $p_{ii} = 1$, then we have a deadlock state and the probability to reach any other state is zero. Therefore, $Prob_M(s_i, \psi) = 0$. Notice also that if for all i, each row of the matrix P' sums up to 1, then for all i, $Prob_M(s_i, \psi) = 0$. Indeed, in this case, we have infinite cycles in S_{maybe} and the system will never reach a state that satisfies ϕ_2 by definition of S_{maybe}. For all other cases, we prove the following proposition.

Proposition 1. *Let A be a square matrix of dimension $n \times n$ such that*

- $\forall i, j, 1 \leq i, j \leq n, a_{ij} \in [0, 1]$
- $\forall i, 1 \leq i \leq n, 0 < \sum_{j=1}^{n} a_{ij} \leq 1$
- $\exists i, 1 \leq i \leq n, \sum_{j=1}^{n} a_{ij} < 1$

Let I_n denote the identity matrix of dimension n. Then the matrix $A - I_n$ is invertible.

Proof. Let $\lambda \in \mathbb{R}^n$ be such that

$$(A - I_n)\lambda = 0 \ .$$

We prove by contradiction that the kernel of $(A - I_n)$ is reduced to 0, that is that $\lambda = 0$ is the only possible solution. Suppose that $\|\lambda\|_\infty > 0$. Suppose further that $\|\lambda\|_\infty = |\lambda_i|$, where

$$\|\lambda\|_\infty \stackrel{\text{def}}{=} \max_{1 \leq i \leq n} |\lambda_i| \ .$$

Part 1: We first prove that i is necessarily such that $\sum_{j=1}^{n} a_{ij} < 1$. Part 2: We then prove a contradiction which makes $\|\lambda\|_\infty = 0$ and ends the proof.

Part 1. Suppose that i is such that

$$\sum_{j=1}^{n} a_{ij} = 1 \ . \tag{5}$$

Then, $(A - I_n)\lambda = 0$ gives for the row i of $(A - I_n)$,

$$\sum_{j=1, j \neq i}^{n} \lambda_j a_{ij} + \lambda_i (a_{ii} - 1) = 0, \tag{6}$$

or equivalently using (5),

$$\sum_{j=1, j \neq i}^{n} (\lambda_j - \lambda_i) a_{ij} = 0 \ .$$

If $\lambda_i > \lambda_j$ for all j, $j \neq i$, and since a_{ij} are non-negative and not all null, then the above equality does not hold. Thus, there exists j such that $\lambda_i \leq \lambda_j$.

Part 2. Recall that $\|\lambda\|_\infty = |\lambda_i|$. We now know that $\sum_{j=1}^{n} a_{ij} < 1$. Using again equation (6), and dividing its both sides by $\|\lambda\|_\infty > 0$, we obtain

$$\sum_{j=1}^{n} \frac{\lambda_j}{\|\lambda\|_\infty} a_{ij} = \frac{\lambda_i}{\|\lambda\|_\infty} \ .$$

We know that

$$\forall j, 1 \leq j \leq n : \frac{|\lambda_j|}{\|\lambda\|_\infty} \leq 1 \ .$$

Therefore, multiplying both sides of the above inequality by $a_{ij} \geq 0$ then summing up all the inequalities we obtain, gives

$$\frac{\lambda_i}{\|\lambda\|_\infty} \leq \sum_{j=1}^{n} a_{ij} < 1,$$

which contradicts the fact that $|\lambda_i| = \|\lambda\|_\infty$. □

Therefore, under the conditions of Proposition 1, the solution is simply given by $v = (I - P')^{-1} b$, where I denotes the identity matrix and $(I - P')^{-1}$ the inverse of the matrix $(I - P')$.

3 Model Checking of Bounded Properties

The straightforward extension of model checking DTMCs to IDTMCs using interval analysis leads to highly imprecise results. We present hereafter our technique to overcome this loss of precision. We focus first on the bounded case, that is $k < +\infty$. The unbounded case is later discussed in Section 4.

3.1 Approximate Model Checking

For an IDTMC, we need to compute the set defined in equation (1). This can be done by replacing the real-valued matrix P' in equation (4) by an interval-valued matrix $\boldsymbol{P'}$ in the computation of the updated component of vector v_k.

The successive computation of each recursion step during the unrolling inherits from the loss of precision due to interval arithmetic (IA) [18]. This can lead to coarse results, sometimes even outside of $[0, 1]$.

To overcome this loss of precision, in the bounded case, we use *affine arithmetic* (AA) [2]. AA was introduced to overcome the loss of relations in interval arithmetic. Consider for instance a symbolic variable v known to be within the interval $[a, b]$. Using IA, an over-approximation of the expression $v - v$ is given by the interval $[a, b] - [a, b]$ = $[a - b, b - a]$, which is a coarse approximation of the actual result $\{0\}$. In AA, the interval $[a, b]$ is represented using the affine expression:

$$\frac{a+b}{2} + \frac{b-a}{2}\epsilon_1,$$

where an error symbol ϵ_1 is introduced to capture an uncertainty within $[-1, 1]$. Now, observe that using AA, we obtain the exact result for the expression $v - v$, that is $\{0\}$. This improvement is due to the fact that the relation between both operands of the subtraction is captured by sharing the same error symbol ϵ_1.

Definition 3 (Affine forms). *An affine form $\hat{a}$ of length l is defined by*

$$\hat{a} \stackrel{\text{def}}{=} \alpha_0^a + \alpha_1^a \epsilon_1 + \cdots + \alpha_l^a \epsilon_l = \alpha_0^a + \sum_{i=1}^{l} \alpha_i^a \epsilon_i,$$

where $\alpha_0^a, \ldots, \alpha_l^a$ are real coefficients, called error weights, and $\epsilon_1, \ldots, \epsilon_l$ are symbolic error variables, known to be within $[-1, 1]$.

AA is closed under linear transformation operations. However, non-linear operations need to be linearized.

Definition 4 (Linear operations). *Let $\hat{a}$ and $\hat{b}$ be two affine forms, let λ, ζ be two finite real numbers, then*

$$\hat{a} \pm \hat{b} \stackrel{\text{def}}{=} (\alpha_0^a \pm \alpha_0^b) + \sum_{i=1}^{l} (\alpha_i^a \pm \alpha_i^b)\epsilon_i$$

$$\lambda \hat{a} \stackrel{\text{def}}{=} \lambda \alpha_0^a + \sum_{i=1}^{l} (\lambda \alpha_i^a)\epsilon_i$$

$$\hat{a} + \zeta \stackrel{\text{def}}{=} (\alpha_0^a + \zeta) + \sum_{i=1}^{l} \alpha_i^a \epsilon_i$$

In the following, we improve the computation of the recurrence equation (4) using both AA and IA. The main idea is to split $\boldsymbol{P}$ into a central matrix P_c, and an interval matrix $\boldsymbol{E}$, which encodes the uncertainty of the model. Note that P_c is a real-valued matrix while $\boldsymbol{E}$ has ranged probabilities. The matrix P_c is built using the centers of the original intervals (which are the means given by the underlying learning technique). All intervals in the uncertainty matrix $\boldsymbol{E}$ are symmetric. Each interval component of $\boldsymbol{E}$, denoted by $[-e_{ij}, e_{ij}]$, is substituted using the symbol ϵ_{ij} known to be within $[-e_{i,j}, e_{i,j}]$. The interval matrix $\boldsymbol{E}$ is then represented by its corresponding affine form matrix $E(\epsilon)$. For each row i of P_c and $E(\epsilon)$ respectively, we assume that:

$$\sum_{j=1}^{n} p_{c_{ij}} = 1 \quad \text{and} \quad \sum_{j=1}^{n} \epsilon_{ij} = 0 \ . \tag{7}$$

These equalities hold for Markov Chain with symmetric uncertainties related to transition probabilities. Usually, statistical techniques (such as bootstrapping [5]) are used to learn a Markov Chain from a (finite) set of observations (logs for instance) of the real system. The uncertainty is related to the required confidence and can be made arbitrarily small using additional observations.

Using the above notations, the recurrence of (4) becomes:

$$v_k(\epsilon) = (P'_c + E'(\epsilon))v_{k-1}(\epsilon) + (b + b(\epsilon)), \tag{8}$$

where P'_c, E', b and $b(\epsilon)$ are derived from P_c and E respectively as detailed in Section 3.1, equation (3). The updated components of the successive iterations of v_k are non-linear (precisely polynomial) functions of the perturbations $(\epsilon_{ij})_{1\leq i,j\leq n}$. To exactly compute the box $\boldsymbol{v_k}$, we therefore need to solve, n instances (one for each row i) of the following (non-linear) optimization problem:

$$\begin{array}{rl} \max / \min & v_{k\,i}(\epsilon) \\ \text{s.t.} & \epsilon_{ij} \in \boldsymbol{\epsilon}_{ij}, 1 \leq i, j \leq n \\ & \displaystyle\sum_{j=1}^{n} \epsilon_{ij} = 0, 1 \leq i \leq n \end{array} \tag{$\star$}$$

To reduce the complexity of the propagation of the components $v_k[i](\epsilon)$, we compute the first-order terms exactly using AA, and over-approximate all high-order terms using IA. Formally, the vector $v_k(\epsilon)$ of equation (8) is reduced from a polynomial function of ϵ_{ij} to a linear function of these perturbations plus an interval which over-approximates the non-linear error terms:

$$v_k(\epsilon) \in \tilde{\boldsymbol{P}}_k \stackrel{\text{def}}{=} c_k + l_k(\epsilon) + \square_k, \tag{9}$$

where c_k is a constant, $l_k(\epsilon)$ is the linear part of $v_k(\epsilon)$, and $\square_k$ is an over-approximation of $v_k(\epsilon) - (c_k + l_k(\epsilon))$. The recurrence of the computation of $\boldsymbol{v}_k$ is now split into three components, c_k, $l_k(\epsilon)$ and $\square_k$, updated as follows:

$$\begin{aligned} c_k &= P'_c c_{k-1} + b \\ l_k(\epsilon) &= P'_c l_{k-1}(\epsilon) + E'(\epsilon)c_{k-1} + b(\epsilon) \\ \square_k &= P'_c \square_{k-1} + \boldsymbol{E'}(\square_{k-1} + \boldsymbol{l}_{k-1}) \end{aligned} \tag{10}$$

The constant c_k is calculated from c_{k-1}, P'_c and b. This calculation gives the probability with which the DTMC defined by P_c satisfies the property.

The elements of the vector $l_k(\epsilon)$ are expressed as a linear combination of elements in $l_{k-1}(\epsilon)$ and the elements of the matrix $E(\epsilon)$. Therefore, each component of the vector $l_k(\epsilon)$ is of the form $\sum_{1\leq i,j\leq n} \alpha_{ij}\epsilon_{ij}$.

To compute $\square_k$, we need to compute the interval vector $\boldsymbol{l}_{k-1}$. Each component of $\boldsymbol{l}_{k-1}$ is the wrapping interval of the affine expression given by the ith element of l_{k-1}. Therefore, at each step, we have to compute the n objective values of the following linear programming problems:

$$\begin{array}{rl} \max / \min & \displaystyle\sum_{1\leq i,j\leq n} \alpha_{ij}\epsilon_{ij} \\ \text{s.t.} & -e_{ij} \leq \epsilon_{ij} \leq e_{ij}, 1 \leq i, j \leq n \\ & \displaystyle\sum_{j=1}^{n} \epsilon_{ij} = 0, 1 \leq i \leq n \end{array} \tag{P}$$

We can use any off-the-shelf LP Solver, such as GLPK [17], to solve (P). However, we present, in the next section, a specific efficient algorithm. We illustrate first all steps detailed earlier in a concrete example.

Example 1. Consider an IDTMC defined by the 4-tuple $\boldsymbol{M} = (S, s_i, \boldsymbol{P}, \ell)$. Suppose that $S = \{s_1, s_2, s_3, s_4\}$, $AP = \{a, b\}$, $s_i = s_1$, $\ell(s_1) = \{b\}$, $\ell(s_2) = \{a\}$, $\ell(s_3) = \{a \wedge b\}$, $\ell(s_4) = \{b\}$, and

$$\boldsymbol{P} = \begin{bmatrix} 0 & [0.49, 0.51] & [0.09, 0.11] & [0.39, 0.41] \\ [0.49, 0.51] & 0 & 0 & [0.49, 0.51] \\ 0 & [0.79, 0.81] & [0.19, 0.21] & 0 \\ [0.49, 0.51] & [0.29, 0.31] & [0.19, 0.21] & 0 \end{bmatrix} .$$

Suppose we want to verify the PCTL property $P_{\leq \gamma}[\psi]$, where $\psi = b\ \mathcal{U}^{\leq 2}(a \wedge b)$. We compute $Prob_{\boldsymbol{M}}(s_i, \psi)$ for all states s_i following the recursion of equation (10). In this example $S_{yes} = \{3\}$, $S_{no} = \{2\}$ and $S_{maybe} = \{1, 4\}$. We first extract the square matrix P_c and the error matrix $E(\epsilon)$:

$$P_c = \begin{bmatrix} 0 & 0.5 & 0.1 & 0.4 \\ 0.5 & 0 & 0 & 0.5 \\ 0 & 0.8 & 0.2 & 0 \\ 0.5 & 0.3 & 0.2 & 0 \end{bmatrix} \text{ and } E(\epsilon) = \begin{bmatrix} 0 & \epsilon_{(1,2)} & \epsilon_{(1,3)} & \epsilon_{(1,4)} \\ \epsilon_{(2,1)} & 0 & 0 & \epsilon_{(2,4)} \\ 0 & \epsilon_{(3,2)} & \epsilon_{(3,3)} & 0 \\ \epsilon_{(4,1)} & \epsilon_{(4,2)} & \epsilon_{(4,3)} & 0 \end{bmatrix}$$

The matrices P'_c, $E'(\epsilon)$ and the vectors b and $b(\epsilon)$ are then given by (equation (3)):

$$P'_c = \begin{bmatrix} 0 & 0.4 \\ 0.5 & 0 \end{bmatrix} \qquad E'(\epsilon) = \begin{bmatrix} 0 & \epsilon_{(1,4)} \\ \epsilon_{(4,1)} & 0 \end{bmatrix}$$
$$b = (0.1, 0.2)^t \qquad b(\epsilon) = (\epsilon_{(1,3)}, \epsilon_{(4,3)})^t \ .$$

In this example, all errors ϵ_{ij}, $1 \leq i, j \leq 4$, are within $[-0.01, 0.01]$. The vector $l_k(\epsilon)$ represents the first-order error as a linear combination of the ϵ_{ij}. The intervals vector $\square_0$ represents an over-approximation of the second and higher-order errors. Both $l_0(\epsilon)$ and $\square_0$ are null. The initial vector $\boldsymbol{v}_0$ is exactly equal to c_0. It is constructed using the probabilities we already know, and by initialization those of S_{maybe} to zero:

$$\boldsymbol{v}_0 = c_0 = [0, 0]^t \ .$$

Following (10), we only update the probabilities of the states of the set S_{maybe} (here the first and the fourth components). We get:

$$c_1 = P'_c \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} + b = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

Similarly for $l_1(\epsilon)$, we get

$$l_1(\epsilon) = P'_c \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} + E'(\epsilon) \times \begin{bmatrix} 0 \\ 0 \end{bmatrix} + b(\epsilon) = \begin{bmatrix} \epsilon_{(1,3)} \\ \epsilon_{(4,3)} \end{bmatrix}$$

Therefore,

$$c_1 = (0.1, 0.2)^t; \quad l_1(\epsilon) = (\epsilon_{(1,3)}, \epsilon_{(4,3)}); \quad \boldsymbol{l}_1 = \begin{bmatrix}[-0.01, 0.01] \\ [-0.01, 0.01]\end{bmatrix}; \quad \square_1 = 0 \ .$$

For the second iteration we obtain:

$$c_2 = \begin{bmatrix}0.18 \\ 0.25\end{bmatrix}; \quad l_2(\epsilon) = \begin{bmatrix}\epsilon_{(1,3)} + 0.2\epsilon_{(1,4)} + 0.4\epsilon_{(4,3)} \\ 0.5\epsilon_{(1,3)} + 0.1\epsilon_{(4,1)} + \epsilon_{(4,3)}\end{bmatrix}; \quad \square_2 = \begin{bmatrix}[-10^{-4}, 10^{-4}] \\ [-10^{-4}, 10^{-4}]\end{bmatrix}$$

Finally, we obtain:

$$\begin{bmatrix}Prob_{\boldsymbol{M}}(s_1, \psi) \\ Prob_{\boldsymbol{M}}(s_2, \psi) \\ Prob_{\boldsymbol{M}}(s_3, \psi) \\ Prob_{\boldsymbol{M}}(s_4, \psi)\end{bmatrix} = \begin{bmatrix}[0.1639, 0.1961] \\ 0 \\ 1 \\ [0.2339, 0.2661]\end{bmatrix} \ .$$

3.2 Bounding the First-Order Error Terms

In problem (P), for each j, the set of constraints involving the variables ϵ_{ij}, $1 \leq i \leq n$ are independent from all other constraints. Therefore, the problem (P) can be equivalently decomposed into n smaller problems (in the worst case) of the form:

$$\begin{aligned} \max / \min \quad & \sum_{1 \leq i \leq n} \alpha_i \epsilon_i \\ \text{s.t.} \quad & -e_i \leq \epsilon_i \leq e_i, 1 \leq i \leq n \\ & \sum_{i=1}^{n} \epsilon_i = 0 \end{aligned} \tag{L}$$

where the interval $[-e_i, e_i] \stackrel{\text{def}}{=} \boldsymbol{\epsilon}_i$. Note that due to the symmetric nature of the feasible region, we see that if the tuple $(\bar{\epsilon}_1, \ldots, \bar{\epsilon}_n)$ maximizes the objective function of L, then the tuple $(-\bar{\epsilon}_1, \ldots, -\bar{\epsilon}_n)$ minimizes it, and vice versa. In the sequel, we focus on the maximization problem.

We start by observing that $(\imath)$ the feasible region is non-empty (as it contains the tuple $(0, \ldots, 0)$), and $(\imath\imath)$ the objective function is bounded, ($\sum_{i=1}^{n} \alpha_i e_i$ being an upper bound). Thus the set of solutions for problem (L) is non-empty. Furthermore, (L) need not have a unique solution in general. Indeed, if all α_is are equal, then all feasible solutions are optimal in that they maximize the objective function.

For a feasible tuple $(\epsilon_1, \ldots, \epsilon_n)$, we say that ϵ_i is *positively* or *negatively saturated* accordingly as ϵ_i equals e_i or $-e_i$, respectively. In order to formulate a linear time algorithm for (L), we exploit the useful fact that there always exists a maximizing feasible solution that saturates (positively or negatively) all but possibly one variable, say ϵ_k. Then the maximization problem reduces to determining the variables that need to be saturated positively and the ones that need to be saturated negatively which in turn automatically determines the values assigned to all the variables ϵ_i. Finally, we show that determining the positively and negatively saturated variables reduces to an instance of the *Weighted Median Problem* which is known to be solvable in linear time [1].

Lemma 1 (Saturation Lemma). *Given a linear programming problem of the form of* (L)*, there exists a feasible maximizing solution that leaves at most one variable non-saturated. All other variables are positively or negatively saturated.*

Proof. Suppose that $\bar{\epsilon}_i$ and $\bar{\epsilon}_j$ are not saturated. Suppose further that $\alpha_i \leq \alpha_j$. We can increase the objective value by

$$(\alpha_i - \alpha_j) \min\{e_i - \bar{\epsilon}_i, e_j - \bar{\epsilon}_j\},$$

if we update the values of ϵ_i and ϵ_j as follows:

$$\begin{aligned} \epsilon_i &= \bar{\epsilon}_i - \min\{e_i - \bar{\epsilon}_i, e_j - \bar{\epsilon}_j\} \\ \epsilon_j &= \bar{\epsilon}_j + \min\{e_i - \bar{\epsilon}_i, e_j - \bar{\epsilon}_j\} \end{aligned}$$

We still have $\epsilon_i \in [-e_i, e_i]$, and $\epsilon_j \in [-e_j, e_j]$. Moreover, $\epsilon_i + \epsilon_j = \bar{\epsilon}_i + \bar{\epsilon}_j$, then the above update is feasible (all constraints are respected). Since $\bar{\epsilon}_i$ and $\bar{\epsilon}_j$ are not saturated, we have

$$\min\{e_i - \bar{\epsilon}_i, e_j - \bar{\epsilon}_j\} \neq 0,$$

hence $\epsilon_i \neq \bar{\epsilon}_i$ and $\epsilon_j \neq \bar{\epsilon}_j$ which contradicts the fact that $(\bar{\epsilon}_1, \ldots, \bar{\epsilon}_n)$ is an optimal solution. □

Following Lemma 1, it turns out that in order to solve (P), it suffices to determine the non-saturated index, say k, as well as the sets $\oplus$ and $\ominus$ of positively and negatively saturated variables, respectively. This, in turn, determines a maximizing feasible assignment to all the variables as follows: If $\epsilon_i \in \ominus$ (resp. $\oplus$), then $\epsilon_i = -e_i$ (resp. e_i). The value of the remaining non-saturated variable ϵ_k is then deduced as follows:

$$\epsilon_k = -\sum_{i \in \ominus \cup \oplus} \epsilon_i = \sum_{i \in \ominus} e_i - \sum_{i \in \oplus} e_i \ . \tag{11}$$

The problem of finding a maximizing feasible solution for (L) now reduces to determining the possibly non-saturated variable ϵ_k which we formulate as an instance of the Weighted Median Problem.

Intuitively this is easy to see as in order to maximize (L) we need to assign as large positive values as possible to the variables ϵ_i, $1 \leq i \leq n$, with the largest coefficients α_i. Due to the constraint $\sum_{i=1}^{n} \epsilon_i = 0$, if some variables are assigned positive values then there will exist others that need to be assigned negative values. These negative values should be assigned to variables ϵ_i with the smallest coefficients α_i. In fact, the *balancing* constraint $\sum_{i=1}^{n} \epsilon_i = 0$ implies that, roughly speaking, the sum of the weights, i.e., values of ϵ_i, of the positively and negatively assigned variables are balanced. This immediately leads to an instance of the Weighted Median Problem as follows. The non-saturated variable ϵ_k can be identified as a solution to the Weighted Median Problem where we associate to each ϵ_i the weight e_i, and look for the weighted median ϵ_k defined by

$$\sum_{\epsilon_i < \epsilon_k} e_i < \frac{1}{2} \sum_{i=1}^{n} e_i \leq \sum_{\epsilon_i \leq \epsilon_k} e_i \ . \tag{12}$$

Following our discussion above, the variables ϵ_i with the largest (resp. smallest) coefficients α_i need to be positively (resp. negatively) saturated. Thus we define the sets $\ominus$ and $\oplus$ as follows: $\ominus \stackrel{\text{def}}{=} \{i \mid \alpha_i < \alpha_k\}$, $\oplus \stackrel{\text{def}}{=} \{i \mid \alpha_i > \alpha_k\}$. Finally we need to show the optimality of the resulting solution. Formally,

Theorem 1 (Optimality Result). *The tuple* $(\bar{\epsilon}_1, \ldots, \bar{\epsilon}_n)$*, where* $\bar{\epsilon}_i = -e_i$ *or* $\bar{\epsilon}_i = e_i$*, accordingly as* ϵ_i *belongs to* $\ominus$ *or* $\oplus$ *, respectively, and* ϵ_k *is defined as in* (11) *is a maximizing feasible solution for* (L)*.*

Proof. We first prove that $(\bar{\epsilon}_1, \ldots, \bar{\epsilon}_n)$ is feasible, that is that $\bar{\epsilon}_k \in [-e_k, e_k]$ (all other conditions are satisfied by construction). Then we prove that the so defined solution is optimal.

Feasibility. By definition of the weighted median value, we have

$$\sum_{i \in \ominus} e_i < \frac{1}{2} \sum_{i=1}^{n} e_i \leq \sum_{\alpha_i \leq \alpha_k} e_i = e_k + \sum_{i \in \ominus} e_i$$

$$\sum_{i \in \oplus} e_i \leq \frac{1}{2} \sum_{i=1}^{n} e_i < \sum_{\alpha_i \geq \alpha_k} e_i = e_k + \sum_{i \in \oplus} e_i$$

Using (11), we subtract the above inequalities. We obtain

$$-e_k + \epsilon_k < 0 \leq e_k + \epsilon_k,$$

which is equivalent to $-e_k \leq \epsilon_k < e_k$.

Optimality. Starting from $(\bar{\epsilon}_1, \ldots, \bar{\epsilon}_n)$, we prove that any update does not improve the objective value reached for this particular configuration we started with.

Suppose that $\bar{\epsilon}_k \leq e_k$. We add a non-negative quantity δ to $\bar{\epsilon}_k$. Since $\sum_{i=1}^{n} \bar{\epsilon}_i = 0$, the quantity δ needs to be subtracted from some $\bar{\epsilon}_i$ such that $\alpha_i > \alpha_k$, indeed all other ϵ_i are saturated to their lowest possible value $-e_i$. Let δ_i denote the amount we subtract from $\bar{\epsilon}_i$ such that $\alpha_i > \alpha_k$. We have $\sum_{\alpha_i > \alpha_k} \delta_i = \delta$ and the new objective value is equal to

$$\sum_{\alpha_i < \alpha_k} \alpha_i \bar{\epsilon}_i + \sum_{\alpha_i > \alpha_k} \alpha_i (\bar{\epsilon}_i - \delta_i) + \alpha_k (\bar{\epsilon_k} + \delta)$$

$$= \sum_{i=1}^{n} \alpha_i \bar{\epsilon}_i + \underbrace{\sum_{\alpha_i > \alpha_k} (\alpha_i - \alpha_k) \delta_i}_{\Delta} \quad . \quad (13)$$

By definition the quantity Δ is non-positive. We conclude that adding a non-negative quantity to ϵ_k does not improve the initial objective value.

By a similar reasoning we prove that subtracting a non-negative quantity from ϵ_k, or updating any other $\bar{\epsilon}_i$ decreases the initial objective value. □

Since the weighted median problem can be solved in linear time [1], the problem (L) has the same complexity.

Proposition 2. *The problem* (L) *can be solved in* $O(n)$.

We now derive the worst-case complexity to compute the over-approximation $\tilde{\boldsymbol{P}}_k$ (see equation (9)).

Proposition 3. *Computing the over-approximation* $\tilde{\boldsymbol{P}}_k$*, defined in equation* (9)*, can be done in* $O(kn^3)$*, where* k *is the iteration depth, and* n *the number of states.*

Proof. For each iteration of $\tilde{\boldsymbol{P}}_i$, $1 \leq i \leq k$, for each line of the square $(n \times n)$ matrix $\tilde{\boldsymbol{P}}_i$ we have to solve in the worst case n instances of (L), which has a linear complexity (see Proposition 2). Therefore, for k iterations, computing $\tilde{\boldsymbol{P}}_k$ requires at most $O(kn^3)$ operations. □

4 Model Checking of Unbounded Properties

For infinite paths, we need to compute the limit of the recursion defined in equation (10).

$$\begin{aligned} c &= P'_c c + b \\ l(\epsilon) &= P'_c l(\epsilon) + E'(\epsilon)c + b(\epsilon) \\ \square &= P'_c \square + \boldsymbol{E}'(\square + \boldsymbol{l}) \end{aligned} \tag{14}$$

According to Proposition 1, the matrix $P'_c - I$ is invertible. Therefore,

$$\begin{aligned} c &= (I - P'_c)^{-1} b \\ l(\epsilon) &= (I - P'_c)^{-1}(E'(\epsilon)c + b(\epsilon)) \end{aligned}$$

The only remaining component to compute is $\square$. If $h(\epsilon)$ denotes the exact high order perturbation of the vector $\boldsymbol{v_k}$, then with respect to equation (4), we obtain:

$$(I - P'_c - E'(\epsilon))h(\epsilon) = E'(\epsilon)l(\epsilon) \ .$$

An over-approximation of $h(\epsilon)$ can be derived as the solution of the following system of interval linear equations:

$$(I - P'_c - \boldsymbol{E'})\square = \boldsymbol{E'l} \tag{15}$$

Such systems were widely studied during the last decades both from a theoretical (solvability and complexity) and practical (implementations and tools) point of views [3,13]. The solvability is proved to be NP-hard. However, using numerical techniques to approximate the set of solutions such as in [10] can be used to efficiently solve the problem.

Example 2. Going back to example 1, we now model check the time-unbounded property $P_{\leq\gamma}[b\ \mathcal{U}(a \wedge b)]$. We have

$$\begin{aligned} c &= (0.225, 0.3125)^t \\ l &= \begin{bmatrix} 0.39\epsilon_{(1,4)} + 1.25\epsilon_{(1,3)} + 0.1125\epsilon_{(4,1)} + 0.5\epsilon_{(4,3)} \\ 0.195\epsilon_{(1,4)} + 0.625\epsilon_{(1,3)} + 0.28\epsilon_{(4,1)} + 1.25\epsilon_{(4,3)} \end{bmatrix} \\ \boldsymbol{l} &= \begin{bmatrix} [-0.0226, 0.0226] \\ [-0.0235, 0.0235] \end{bmatrix} \end{aligned}$$

Finally $\square$ is the solution of the following system

$$-\begin{bmatrix} -1 & [0.39, 0.41] \\ [0.49, 0.51] & -1 \end{bmatrix} \square = \begin{bmatrix} [-0.00235, 0.00235] \\ [-0.00226, 0.00226] \end{bmatrix}$$

Using `PROFIL/BIAS` [19], we obtain

$$\square = ([-0.004149, 0.004149]; [-0.00438, 0.00438])^t$$

Which makes

$$\begin{bmatrix} Prob_{\boldsymbol{M}}(s_1, \psi) \\ Prob_{\boldsymbol{M}}(s_2, \psi) \\ Prob_{\boldsymbol{M}}(s_3, \psi) \\ Prob_{\boldsymbol{M}}(s_4, \psi) \end{bmatrix} = \begin{bmatrix} [0.1973, 0.2526] \\ 0 \\ 1 \\ [0.2855, 0.3395] \end{bmatrix} .$$

Observe that

$$\boldsymbol{P'v} \subseteq \boldsymbol{v} \ .$$

5 Case Study

We applied our approach to model check a smart grid management system. We analyzed a model consisting of 21 states, where each state corresponds to a range of the difference (δ) between energy supply and demand. We used our approximate model checking technique to check the following two properties (δ_m and δ_M represent the minimum and maximum values of δ respectively):

Table 1. IA versus AA+LP

	# Days	IA	AA+LP
P_1	7	$[0.55, 1]$	$[0.83, 0.98]$
P_2	7	$[0.35, 1]$	$[0.70, 0.80]$

P_1: What is the probability that within k days, the power grid will switch from high supply mode to low supply mode: $P[\frac{1}{2}\delta_M \leq \delta \leq \delta_M \ \mathcal{U}^{\leq k} 0 \leq \delta \leq \frac{1}{2}\delta_M]$.

P_2: What is the probability that within k days, the power grid will switch from low supply mode to low demand mode: $P[0 \leq \delta \leq \frac{1}{2}\delta_M \ \mathcal{U}^{\leq k} \frac{1}{2}\delta_m \leq \delta \leq 0]$.

As can be seen in Table 1, using our approach based on affine arithmetic, we are able to compute a much tighter probability range for the two properties with negligible computation overhead (all computation took less than 0.1 second).

6 Conclusion

In this paper we have presented a new technique to address the problem of computing the probability of a reachability property. We leverage affine and interval arithmetic to propagate the uncertainties of the learned transition probabilities. At each step of the model checking procedure, the first-order error terms are computed optimally in linear time via a reduction to the weighted median problem. As a future avenue, we plan to investigate model checking of properties with multiple P operators.

References

1. Bleich, C., Overton, M.L.: A linear-time algorithm for the weighted median problem. Courant Institute of Mathematical Sciences, New York University, New York (1983)
2. Comba, J.L.D., Stolfi, J.: Affine arithmetic and its applications to computer graphics. In: SIBGRAPI 1993 (1993)
3. Corsaro, S., Marino, M.: Interval linear systems: the state of the art. Computational Statistics 21, 365–384 (2006)
4. de Figueiredo, L.H., Stolfi, J.: Self-Validated Numerical Methods and Applications. Brazilian Mathematics Colloquium monographs. IMPA/CNPq, Rio de Janeiro, Brazil (1997)
5. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. CRC Press (1993)
6. Fecher, H., Leucker, M., Wolf, V.: *Don't Know* in Probabilistic Systems. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 71–88. Springer, Heidelberg (2006)
7. Ghorbal, K., Goubault, E., Putot, S.: A Logical Product Approach to Zonotope Intersection. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 212–226. Springer, Heidelberg (2010)
8. Girard, A.: Reachability of Uncertain Linear Systems Using Zonotopes. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 291–305. Springer, Heidelberg (2005)
9. Goubault, É., Putot, S.: Static Analysis of Numerical Algorithms. In: Yi, K. (ed.) SAS 2006. LNCS, vol. 4134, pp. 18–34. Springer, Heidelberg (2006)
10. Hansen, E., Sengupta, S.: Bounding solutions of systems of equations using interval analysis. BIT Numerical Mathematics 21, 203–211 (1981)
11. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994)
12. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate Probabilistic Model Checking. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 73–84. Springer, Heidelberg (2004)
13. Jiri, Rohn: Systems of linear interval equations. Linear Algebra and its Applications 126, 39–78 (1989)
14. Jonsson, B., Larsen, K.: Specification and refinement of probabilistic processes. In: LICS, pp. 266–277 (July 1991)
15. Katoen, J.-P., Klink, D., Leucker, M., Wolf, V.: Three-valued abstraction for probabilistic systems. Journal of Logic and Algebraic Programming 81(4), 356–389 (2012)
16. Kozine, I.O., Utkin, L.V.: Interval-valued finite markov chains. Reliable Computing 8, 97–113 (2002)
17. Makhorin, A.: The GNU Linear Programming Kit (GLPK) (2000)
18. Moore, R.E., Yang, C.T.: Interval analysis I. Technical Report LMSD-285875, Lockheed Missiles and Space Division, Sunnyvale, CA, USA (1959)
19. Rump, S.M.: Profil/bias
20. Sen, K., Viswanathan, M., Agha, G.: On Statistical Model Checking of Stochastic Systems. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 266–280. Springer, Heidelberg (2005)
21. Younes, H.L.S., Simmons, R.G.: Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 223–235. Springer, Heidelberg (2002)
22. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to Simulink/Stateflow verification. In: HSCC, pp. 243–252 (2010)