LTL model checking of Interval Markov Chains

Michael Benedikt, Rastislav Lenhardt, and James Worrell

Department of Computer Science, University of Oxford, United Kingdom

Abstract. Interval Markov chains (IMCs) generalize ordinary Markov chains by having interval-valued transition probabilities. They are useful for modeling systems in which some transition probabilities depend on an unknown environment, are only approximately known, or are parameters that can be controlled. We consider the problem of computing values for the unknown probabilities in an IMC that maximize the probability of satisfying an ω-regular specification. We give new upper and lower bounds on the complexity of this problem. We then describe an approach based on an expectation maximization algorithm. We provide some analytical guarantees on the algorithm, and show how it can be combined with translation of logic to automata. We give experiments showing that the resulting system gives a practical approach to model checking IMCs.

1 Introduction

Interval Markov chains (IMCs) generalize ordinary Markov chains by allowing undetermined transition probabilities that are constrained to intervals [14]. IMCs arise naturally in the modelling and verification of probabilistic systems. For example, some transition probabilities may depend on an unknown environment, may only be approximately known, or may be parameters that can be optimized.

Interval Markov chains can be seen as a type of Markov decision process. Valuations of their undetermined transition probabilities can correspondingly be seen as history-free stochastic schedulers. This enforced history-independence makes the theory of IMCs different from that of MDPs. In this paper we consider the problem of computing the optimal (either maximum or minimum) probability that an IMC can satisfy some target specification, where the latter is given as an automaton or as a Linear Temporal Logic (LTL) formula. In previous work on verifying IMCs, Chatterjee *et al.* [7] focus on branching-time properties and Delahaye *et al.* [10] consider refinement. While [7] obtain a 2EXPTIME bound for LTL as a consequence of their results, algorithms and complexity bounds for basic linear-time problems on IMCs have, to the best of our knowledge, not been studied in their own right.

We begin with a study of the complexity of optimizing IMCs with respect to linear-time specifications. We give new upper bounds on the reachability problem and the model checking problem for deterministic automata, unambiguous automata, and LTL. We also show that the 2EXPTIME upper-bound from [7] for LTL can be improved to EXPSPACE in general and to PSPACE when the number of parameters is fixed. We complement this with new lower-bounds, showing

that solving the optimization problem for unambiguous automata within the polynomial hierarchy would have significant consequences for the complexity of fundamental problems in symbolic computation

We then turn to practical algorithms for LTL model-checking of IMCs. We use the *expectation-maximization* procedure, which is ubiquitous in machine learning. Indeed, our algorithm can be seen as a variant of the classical Baum-Welch procedure, which finds the optimal probability that an IMC generates a fixed set of sample data. The Baum-Welch procedure progressively re-estimates values of the parameters, giving relatively greater weight to transitions that occur frequently on computations that satisfy a desired property. Analogously with Baum-Welch, we show that our algorithm converges, but not necessarily to the value of the optimal parameters. Our solution to LTL model checking of IMCs couples the expectation-maximization algorithm with a translation of LTL to unambiguous automata. We show that the approach works well in practice, and allows one to take-advantage of the use of unambiguous automata as an intermediate representation.

In summary, our contributions are: (i) Improved upper bounds for model-checking of IMCs with respect to linear-time problems; (ii) New lower bounds, which give new insight into the expressiveness of IMCs; (iii) A novel algorithmic approach to solving the model checking problem in practice; (iv) Experimental results comparing both our LTL translation methods and our end-to-end solution to other techniques. For space reasons, some proofs are omitted.

2 Definitions

Logic and Automata. We specify ω-regular properties using *Linear Temporal Logic* LTL and *Büchi automata*. The formulas of LTL are built from atomic propositions using Boolean connectives and the temporal operators $\bigcirc$ (*next*), $\mathcal{U}$ (*until*) and $\mathcal{R}$ (*release*). Formally, LTL is defined by the following grammar:

$$\varphi ::= p_i \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \,\mathcal{U}\, \varphi \mid \varphi \,\mathcal{R}\, \varphi \mid \bigcirc \varphi\,,$$

where $p_0, p_1, \ldots$ are propositional variables. We abbreviate $\text{true}\,\mathcal{U}\,\varphi$ as $\Diamond\varphi$ and write $\Box\varphi$ for $\neg\Diamond\neg\varphi$. We refer the reader to [17] for the semantics of LTL.

A *generalized Büchi automaton* $\mathcal{A}$ is a tuple $(\Sigma, Q, Q_0, \Delta, \mathcal{F})$ with alphabet Σ, set of states Q, set of initial states $Q_0 \subseteq Q$, transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a collection of accepting sets $\mathcal{F} = \{F_1, \ldots, F_k\}$, where $F_i \subseteq Q$. An infinite run of $\mathcal{A}$ is *accepting* if each set $F \in \mathcal{F}$ is visited infinitely often in the run. We say that $\mathcal{A}$ is *unambiguous* if each word has at most one accepting run.

Interval Markov Chains. A Markov chain is a tuple $\mathcal{M} = (S, \pi_0, M)$, where S is a finite set of states, π_0 is the initial-state distribution on S, and $M : S \times S \to [0,1]$ is a stochastic transition matrix, i.e., $\sum_{t \in S} M(s,t) = 1$ for all $s \in S$. $\mathcal{M}$ induces a Borel probability measure $Pr_{\mathcal{M}}$ on S^ω in the standard way. An *interval Markov chain* is a tuple $\mathcal{M} = (S, \pi_0, M_l, M_u)$ in which the transition matrix of a Markov chain is replaced with two matrices $M_u, M_l : S \times S \to [0,1]$ with $M_l \leq M_u$. Intuitively M_l and M_u give respective lower and upper bounds

on the transition probabilities. An *incomplete Markov chain* is a special case of an interval Markov chain in which for each pair of states s, t, either $M_l(s,t) = M_u(s,t)$ or $M_l(s,t) = 0$ and $M_u(s,t) = 1$, that is, a probability is either precisely given or completely unspecified. An interval Markov chain $\mathcal{M} = (S, \pi_0, M_l, M_u)$ is *refined* by a Markov chain $\mathcal{M}' = (S, \pi_0, M)$ if $M_l(s,t) \leq M(s,t) \leq M_u(s,t)$ for all pairs of states $s, t \in S$. Note that $\mathcal{M}'$ has the same set of states and initial distribution as $\mathcal{M}$.

Given an interval Markov chain $\mathcal{M}$ with set of states S and a labelling function $V : S \to \Sigma$, we want to compute a Markov chain refining $\mathcal{M}$ that optimizes the probability of satisfying a given ω-regular property $L \subseteq \Sigma^\omega$. We call this the *IMC model checking problem*. We will focus in this paper on the case of maximizing the probability of L, but it is easy to modify the techniques to get minimisation. When investigating the complexity of this problem, we will deal with the corresponding decision problem: whether the optimal probability is above a given rational threshold. Let us also note immediately that the problem can be simplified, without loss of generality, by assuming that $\Sigma = S$ and that V is the identity function, i.e., that L is an ω-regular set of trajectories of the IMC. We can do this because ω-regular languages are closed under inverse images of alphabet renamings. We will also use the term *qualitative model checking problem* to refer to the question of whether the probability to satisfy the property L can be made 1.

Product Construction. Next we recall the product construction for Markov chains with unambiguous Büchi automata, which has been noted in several prior works (see, e.g., [8]). An advantage of working with unambiguous automata rather than deterministic automata is that there is a singly exponential translation of LTL to unambiguous automata, whereas the translation of LTL to deterministic automata is doubly exponential.

Let $\mathcal{M} = (S, \pi_0, M)$ be a Markov chain and $\mathcal{A} = (S, Q, Q_0, \Delta, F)$ an unambiguous Büchi automaton whose input alphabet is the set S of states of $\mathcal{M}$. Define the *product graph* $\mathcal{G}_{\mathcal{M}\otimes\mathcal{A}} = (V, E)$ to have set of vertices $V = S \times Q$ and set of edges $E = \{((s,q),(s',q')) : M(s,s')\rangle 0 \text{ and } (q,s',q') \in \Delta\}$.

A strongly connected subset C of $\mathcal{G}_{\mathcal{M}\otimes\mathcal{A}}$ is said to be *accepting* if: (i) for each vertex $(s,q) \in C$, s lies in a bottom strongly connected component of $\mathcal{M}$; (ii) for each vertex $(s,q) \in C$ and edge (s,s') in $\mathcal{M}$ there exists an edge (q,s',q') in $\mathcal{A}$ with $(s',q') \in C$; (iii) for each accepting set $F \in \mathcal{F}$ there exists a vertex $(s,q) \in C$ such that $q \in F$. By extension, a vertex of $\mathcal{G}_{\mathcal{M}\otimes\mathcal{A}}$ is said to be *accepting* if it lies in an accepting set. A vertex is said to be *dead* if it has no path to an accepting vertex. Write V_{yes} for the set of accepting vertices, V_{no} for the set of dead vertices, and $V_?$ for the remaining vertices. Finally, we say that an infinite path in V^ω is *accepting* if it has a tail consisting exclusively of accepting vertices.

We can define probabilistic transitions on the product graph, with a transition from (s,q) to (s',q') being given the value:

$$M'((s,q),(s',q')) = \begin{cases} M(s,s') & ((s,q),(s',q')) \in E \text{ and } (s',q') \in V_{yes} \cup V_? \\ 0 & \text{otherwise} \end{cases}$$

We also define an initial probability vector $\pi'_0 \in \mathbb{R}^V$ by

$$\pi'(s,q) = \begin{cases} \pi_0(s) & (p,s,q) \in \Delta \text{ for some } p \in Q_0 \\ 0 & \text{otherwise} \end{cases}$$

Since $\mathcal{A}$ is non-deterministic, M' need not correspond to a stochastic matrix and π' need not be a probability distribution. Nevertheless M' and π' induce a Borel sub-probability measure $Pr_{\mathcal{M}\otimes\mathcal{A}}$ on V^ω by defining

$$Pr_{\mathcal{M}\otimes\mathcal{A}}(C(v_1 \dots v_n)) = \pi'(v_1) \cdot M'(v_1,v_2) \cdot M'(v_2,v_3) \cdots M'(v_{n-1},v_n)$$

where $C(v_1 \dots v_n)$ is the *cylinder* set of words in V^ω with prefix $v_1 \dots v_n$.

Write $\Diamond V_{yes} \subseteq V^\omega$ for the set of infinite paths that contain an accepting vertex. The following result allows us to reduce the model checking problem for $\mathcal{M}$ and $\mathcal{A}$ to calculating the probability of reaching an accepting vertex in the product graph, which can be done using linear algebra. We can then verify that: $Pr_{\mathcal{M}\otimes\mathcal{A}}(\Diamond V_{yes}) = Pr_{\mathcal{M}}(L(\mathcal{A}))$.

3 Complexity of Verification Problems

Reachability for IMCs is more involved than for MDPs since the interval constraints preclude restricting to deterministic schedulers. As with MDPs we can reduce reachability to linear programming. The resulting linear program is exponential in the size of the IMC, but it has a polynomial-time separation oracle and can therefore be solved in polynomial time using the ellipsoid method.

Proposition 1. *Reachability in interval Markov chains is P-complete.*

Proof. The lower bound is via reduction from the monotone circuit value problem, with the argument identical to P-hardness of computing optimal strategies for reachability in MDPs [16]. The lower bound holds even for incomplete Markov chains.

For the upper bound, we reason as follows. Let $\mathcal{M} = (S, \pi_0, M_l, M_u)$ be an IMC. We can identify the set of refinements of $\mathcal{M}$ with the convex set $[\![\mathcal{M}]\!] \subseteq \mathbb{R}^{S\times S}$ of stochastic transition matrices M such that $M_l(s,t) \le M(s,t) \le M_u(s,t)$ for all $s,t \in S$. Observe that M is a vertex of $[\![\mathcal{M}]\!]$ if and only if for each state s at most one of the outgoing transition probabilities $M(s,t)$ is strictly between its lower bound $M_l(s,t)$ and its upper bound $M_u(s,t)$.

Consider the following linear program with variables $\boldsymbol{x} = \{x_s : s \in S\}$.

$$\begin{array}{l} \text{minimise } \sum_{s\in S} x_s \\ \text{subject to} \\ \boldsymbol{x} \ge M\boldsymbol{x} \quad \text{for each vertex } M \text{ of } [[\mathcal{M}]] \\ \boldsymbol{x} \ge \chi_F \end{array}$$

where χ_F is the characteristic vector of the set F.

By convexity we observe that $\boldsymbol{x} \ge \chi_F$ is feasible for the above program if and only if $\boldsymbol{x} \ge M\boldsymbol{x}$ for all transition matrices $M \in [[\mathcal{M}]]$. Thus $\boldsymbol{x}$ is feasible if and

only if x_s is an upper bound for the probability to reach F from state s for all refinements of $\mathcal{M}$. We conclude that the optimal solution of the linear program gives the maximum probability to reach F over all refinements.

Although the number of constraints in this linear program is exponential in the number of states of $\mathcal{M}$, we do not need their explicit representation. We can use the Ellipsoid algorithm [13] to find the optimal values of x_s in polynomial time. The Ellipsoid algorithm needs an oracle to determine whether given values of x_s are feasible, and, if not, output a separating hyperplane, i.e., the inequality that does not hold. In fact, given a family of values x_s it suffices to consider a single "dominating" constraint in the above program, namely the transition matrix M that simultaneously maximises each entry of $M\boldsymbol{x}$. This matrix is easy to compute: Let $s_1, s_2, \ldots$ be an enumeration of S such that $x_{s_1} \geq x_{s_2} \geq \ldots$. Now for each state $s \in S$, choose $M(s, s_1)$ as high as possible (compatible with all other edges achieving their lower bounds); if $M(s, s_1)$ is at its upper bound, we set $M(s, s_2)$ as high as possible, etc. □

We now turn to verification of properties given as unambiguous Büchi automata. Note that we can not apply a product construction to reduce to the reachability problem for IMCs; the natural product would have the same variable repeated many times in the product chain, introducing correlation. We introduce a practical technique for addressing this problem in the following section. Still, we can get a polynomial space upper bound by reduction to the decision problem for the existential theory of the reals [6].

Theorem 1. *The model-checking problem for unambiguous Büchi automata on IMCs is in PSPACE.*

A matching PSPACE lower bound in Theorem 1 would imply PSPACE-hardness of the decision problem for the existential theory of the reals, which is open. However we can precisely characterise the complexity of the model checking problem for IMCs against unambiguous Büchi automata in terms of the Blum-Shub-Smale (BSS) model of computation over the real field with order $(\mathbb{R}, \leq)$ [4]. In this model each tape cell of a Turing machine can hold a single real number and a decision problem is a language $L \subseteq \mathbb{R}^*$. Arithmetic operations and sign tests have unit cost regardless of the operands, otherwise the classes of polynomial-time problems, denoted $\mathrm{P}_{\mathbb{R}}$, and non-deterministic polynomial-time problems, denoted $\mathrm{NP}_{\mathbb{R}}$, are defined analogously with the classical case. Note that in the definition of $\mathrm{NP}_{\mathbb{R}}$ the "certificate" is a polynomial-length string of real numbers. We now show:

Theorem 2. *The model checking problem for interval Markov chains with respect to unambiguous automata is $NP_{\mathbb{R}}$-complete.*

Proof. The upper bound is easy given that $\mathrm{NP}_{\mathbb{R}}$ allows the guessing of real numbers, which is precisely what is needed in the model checking problem. The lower bound is via reduction from the problem $(0,1)$-Pos, whose input consists of a real polynomial f and threshold $\theta \in (0,1)$, the output being yes iff there

exist values for the variables of f lying in the open interval $(0,1)$ such that $f \geq \theta$. We assume that f is presented as a sum of products of constants $\alpha \in (0,1)$ and *literals* $x, 1-x$, where x is a variable. $(0,1)$-Pos can be shown hard for $\mathrm{NP}_{\mathbb{R}}$ via reduction from the known hard problem of determining whether a polynomial of degree at most 4 has a real root. We first give a brief overview of the reduction from $(0,1)$-Pos to IMC model checking.

Given an instance f, θ of $(0,1)$-Pos, we build an IMC $\mathcal{M}$ with nodes corresponding to constants and variables of f, along with nodes that designate whether a variable x is to be complemented (transformed into $1-x$). We also build a regular expression E so that the probability E can take over $\mathcal{M}$ as a function of the variables of $\mathcal{M}$ corresponds exactly to f. Then the problem $f \geq \theta$ translates to the problem of model checking E on $\mathcal{M}$.

Let $f \geq \theta$ be an instance of $(0,1)$-Pos, where f mentions real constants $\alpha_1, \ldots, \alpha_m$ and variables $x_1, \ldots, x_n$. We derive an incomplete Markov chain $\mathcal{M} = (S, \pi_0, M)$ from f as follows. The set of states is $S = \{c_1, \ldots, c_{n+m}\} \cup \{h, t\}$, with initial distribution π_0 the uniform distribution on $\{c_1, \ldots, c_{m+n}\}$. We think of each state c_i as a biased coin that represents either a constant or a variable. States $c_1, \ldots, c_m$ represent the constants, and accordingly we define fixed transition probabilities $M(c_i, h) = \alpha_i$ and $M(c_i, t) = 1 - \alpha_i$ for $1 \leq i \leq m$. States $c_{m+1}, \ldots, c_{m+n}$ represent the variables, and we leave the transition probabilities $M(c_i, h)$ and $M(c_i, t)$ undefined. We define $M(h, c_i) = M(t, c_i) = 1/(n+m)$ for all $1 \leq i \leq n+m$. All other transition probabilities are zero.

We define a mapping φ from the constants and literals occurring in f to edges of $\mathcal{M}$ by $\varphi(\alpha_i) = c_i h$, $\varphi(x_i) = c_{i+m} h$, and $\varphi(1 - x_i) = c_{i+m} t$. Write $f = \sum_{i=1}^{k} \prod_{j=1}^{l} f_{i,j}$, where each $f_{i,j}$ is a constant or literal. (We can assume that each product has the same number of terms l by suitable padding.) Then we define a regular expression $E = \sum_{i=1}^{k} \prod_{j=1}^{l} \varphi(f_{i,j})$ over alphabet S, the set of states of $\mathcal{M}$. We can further identify a Markov chain $\mathcal{M}'$ refining $\mathcal{M}$ with a valuation of the variables occurring in f, where variable x_i gets the transition probability p_i to go from c_i to h. Under this identification it is easy to see that

$$Pr_{\mathcal{M}'}(ES^{\omega}) = \frac{f(p_1, \ldots, p_n)}{(n+m)^{l+1}} .$$

This equation straightforwardly allows us to reduce a positivity query on f to a model checking query on $\mathcal{M}$. Note that the requirement in $(0,1)$-Pos that variables only take values in $(0,1)$ can be enforced by modifying the specification language to contain only strings with infinitely many occurrences of $c_i h$ for every i. Finally, it is straightforward to represent the specification language as a deterministic automaton of polynomial size. This completes the proof of Theorem 2. □

The classes $\mathrm{P}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{R}}$ can be compared to classical Boolean complexity classes by considering their *Boolean parts*. The Boolean part of a complexity class $\mathcal{C}$ in the BSS model is defined to be $\mathrm{BP}(\mathcal{C}) = \{L \cap \{0,1\}^* : L \in \mathcal{C}\}$. It is well known that NP is contained in $\mathrm{BP}(\mathrm{NP}_{\mathbb{R}})$ [4] and that PosSLP is contained in

$\mathrm{BP}(\mathrm{P}_{\mathbb{R}})$ [2]. (Recall that PosSLP is the problem of determining whether an arithmetic circuit with integer inputs evaluates to a positive number [2]) It follows from Theorem 2 that the model checking problem for IMCs against unambiguous Büchi automata is both NP-hard and PosSLP-hard. The NP lower bound is already known in the form of NP-hardness of the maximum-likelihood problem for hidden Markov chains [1].

Finally, we turn to LTL model-checking. Formerly, the only known upper bound for LTL model-checking of IMCs was 2EXPTIME [7], the same as for general MDPs. Below we note that a better bound of EXPSPACE can be obtained. More interestingly, if the number of parameters is fixed, the complexity reduces to PSPACE.

Theorem 3. *The LTL model checking problem for IMCs is in EXPSPACE, and is PSPACE-hard. For fixed parameters, the problem is PSPACE-complete. The qualitative model-checking problem is PSPACE-complete.*

Proof. We consider interval Markov chains with a fixed number k of undetermined transition probabilities. We represent these probabilities by variables $x_1, \ldots, x_k$ and work with the field of rational functions $\mathbb{F} = \mathbb{Q}(x_1, \ldots, x_k)$.

Let $\mathcal{M}$ be an interval Markov chain and φ an LTL formula with respective sizes $||\mathcal{M}||$ and $||\varphi||$. Using polynomial space in $||\mathcal{M}||$ and $||\varphi||$ one can translate φ into an equivalent unambiguous Büchi automaton $\mathcal{A}$, build the product graph $\mathcal{G}_{\mathcal{M}\otimes\mathcal{A}}$, and derive a corresponding system of linear equations with coefficients in $\mathbb{F}$ whose solution is an element of $\mathbb{F}$ that represents $P_{\mathcal{M}}(L(\mathcal{A}))$ as a function of $x_1, \ldots, x_k$. This system of equations has size exponential in $||\mathcal{M}||$ and $||\varphi||$.

Now systems of linear equations with coefficients in $\mathbb{F}$ can be solved in polylogarithmic space [5]. Thus, using polynomial space in $||\mathcal{M}||$ and $||\varphi||$ overall, we can compute a rational function $f(x_1, \ldots, x_k) \in \mathbb{F}$ that represents the probability $P_{\mathcal{M}}(L(\mathcal{A}))$. Again, the expression representing f has size exponential in $||\mathcal{M}||$ and $||\varphi||$. Finally we use the polylogarithmic-space procedure of Ben-Or, Kozen and Reif [3] for deciding satisfiability of quantifier-free formulas in the first-order theory of real-closed fields over the fixed set of variables $x_1, \ldots, x_k$. With this procedure we can test the existence of transition probabilities $x_1, \ldots, x_k$ such that $P_{\mathcal{M}}(L(\mathcal{A}))$ is greater than a given threshold using overall space that is polynomial in $||\mathcal{M}||$ and $||\varphi||$. □

4 Expectation Maximization Algorithm

In this section we describe an expectation maximization algorithm that, given an initial refinement $\mathcal{M}_0$ of an IMC $\mathcal{M}$, produces a sequence of refinements having successively higher probabilities of satisfying a given ω-regular property, presented as an unambiguous Büchi automaton $\mathcal{A}$. We assume initially that $\mathcal{M}$ is an incomplete Markov chain and discuss the more general case of interval Markov chains later. We also defer until later a discussion of how the initial refinement is chosen.

Overview. Figure 1 gives an outline of the algorithm. The input includes a parameter n governing the number of iterations of the update procedure. The intuitive idea of the update procedure is to assign relatively greater weight to transitions that are most likely to be taken in computations of the current refinement $\mathcal{M}_i$ that are accepted by $\mathcal{A}$.

Algorithm EM
Input: Incomplete Markov chain $\mathcal{M} = (S, \pi_0, M_l, M_u)$, Initial refinement $\mathcal{M}_0$
Unambiguous Büchi automaton $\mathcal{A} = (S, Q, Q_0, \Delta, \mathcal{F})$, Iteration parameter n
Begin
For $i = 0$ to $n - 1$ do
  $\mathcal{M}_{i+1} := update(\mathcal{M}_i)$
End

Fig. 1. EM Algorithm

The Update Procedure. We now explain in more detail the operation of the update procedure. Assume that we are given a refinement $\mathcal{M}_i$ of $\mathcal{M}$ with associated product graph $\mathcal{G}_{\mathcal{M}_i \otimes \mathcal{A}} = (V, E)$. Write M_i for the transition matrix of $\mathcal{M}_i$ and write π_0' and M_i' for the lifting of the initial state distribution and transition matrix of $\mathcal{M}_i$ to the product $\mathcal{G}_{\mathcal{M}_i \otimes \mathcal{A}}$, as defined in Section 2. Given an infinite path $v_1 v_2 v_3 \ldots \in V^\omega$, say that $v_1 \ldots v_n$ is a *minimum accepting prefix* if $v_i \in V_?$ for $1 \leq i \leq n-1$ and $v_n \in V_{yes}$, i.e., v_n is the first accepting vertex on the path.

Write $U \subseteq S \times S$ for the set of pairs (s,t) of states of the incomplete Markov chain $\mathcal{M}$ whose transition probability is undetermined. For each pair of Markov-chain states $(s,t) \in U$ we define a random variable $Z_{s,t} : V^\omega \to \mathbb{N}$ that takes value 0 on any non-accepting path in $\mathcal{G}_{\mathcal{M}_i \otimes \mathcal{A}}$ and otherwise equals the number of occurrences of edge (s,t) in the projection onto $\mathcal{M}$ of the minimum accepting prefix of the path. The update procedure is based on computing $\mathbf{E}[Z_{s,t}]$.

For each pair $(s,t) \in U$ we compute $\mathbf{E}[Z_{s,t}]$ using a variant of the classical *forward-backward* algorithm for hidden Markov models. Given a vertex $(s,q) \in V_?$, define $\alpha(s,q)$ to be the expected value of the random variable that maps each non-accepting path of $\mathcal{G}_{\mathcal{M}_i \otimes \mathcal{A}}$ to 0 and maps each accepting path to the number of occurrences of (s,q) in a minimum accepting prefix of the path. We can compute $\alpha(s,q)$ as the solution to the following system of linear equations:

$$\alpha(s,q) = \begin{cases} \pi_0'(s,q) + \displaystyle\sum_{(t,p) \in V} (\alpha(t,p) + 1)\, M_i'((t,p),(s,q)) & (s,q) \in V_? \\ 0 & (s,q) \notin V_? \end{cases}$$

We further define $\beta(s,q)$ to be the probability to reach an accepting state in $\mathcal{G}_{\mathcal{M}_i \otimes \mathcal{A}}$ starting at state (s,q). We can compute $\beta(s,q)$ as the solution to the

following system of linear equations:

$$\beta(s,q) = \begin{cases} \sum_{(t,p)} M_i'((s,q),(t,p))\,\beta(t,p) & (s,q) \in V_? \\ 1 & (s,q) \in V_{yes} \\ 0 & (s,q) \in V_{no} \end{cases}$$

Finally for each pair $(s,t) \in U$ we define

$$\mathbf{E}[Z_{s,t}] = \sum_{p \in Q} \sum_{q \in Q} \alpha(s,p)\, M_i'((s,p),(t,q))\, \beta(t,q)\,.$$

Furthermore, for a given state $s \in S$, define $\mu_s = \sum\{M_i(s,t) : (s,t) \notin U\}$ to be the total mass of all fixed transition probabilities at state s. The update procedure assigns to $(s,t) \in U$ the transition probability

$$(1-\mu_s) \cdot \frac{\mathbf{E}[Z_{s,t}]}{\sum_{u:(s,u)\in U} \mathbf{E}[Z_{s,u}]} \tag{1}$$

if $(s,q) \notin V_{no}$ for some $q \in Q$. If $(s,q) \in V_{no}$ for all $q \in Q$ then the weight of each edge $(s,t) \in U$ is left unchanged. Thus the new transition probability is determined by the proportion of times that an accepting trajectory of the Markov chain takes the edge (s,t) among all visits to state s before reaching an accepting state of the product.

The following result is proven in the full version.

Theorem 4. *The sequence $Pr_{\mathcal{M}_i}(L(\mathcal{A}))$ is monotonically increasing.*

The choice of initial refinement $\mathcal{M}_0$ can have a significant effect on the behaviour of the EM algorithm. In particular, the choice of which transition values in $\mathcal{M}_0$ are positive governs the initial classification of vertices of the product graph as either accepting or dead. Note that successive iterations of the update procedure do not alter the set of dead vertices. Of course, the connectivity properties of $\mathcal{M}_0$ may be independent of the undefined transition values in an incomplete Markov chain. Furthermore, the choice of positive transitions in $\mathcal{M}_0$ may be suggested by the problem instance. For example, in a repair problem we start with a Markov chain $\mathcal{M}$ and see if it can be made to satisfy a given property by optimizing its transition values within certain intervals. In this case it is natural to take $\mathcal{M}$ itself as the initial refinement. In general, to gain confidence that we have reached the global optimum, we can employ standard heuristics, such as random restart.

We have described the algorithm for incomplete Markov chains, and in the full version we prove that it progressively improves the expectation and converges to a local maximum. For IMCs an update may violate the restricted ranges for intervals. In this case we consider the output of the update algorithm for every refinement of the IMC formed by fixing some parameters to be at the boundary. At least one of these must be feasible (e.g. those where all parameters are fixed), and we choose the refinement that gives the optimal probability.

LTL to Unambiguous Büchi Automata. To apply the expectation maximization algorithm to LTL formulas, we use a translation from LTL to unambiguous generalized Büchi automata. Our approach is a modification of the build-by-need construction of Gerth *et al.* [12] in which we adjust the translation rules from Section 3.2 of [12] to remove potential ambiguity. For example, one step of the [12] procedure involves splitting an automaton state labelled with the subformula $\varphi \vee \psi$ into two copies, one labelled φ and the other labelled ψ. In our approach such a state is instead split into a copy labelled φ and a copy labelled $\neg\varphi \wedge \psi$. The mutual exclusivity of the logical formulas leads to the production of an unambiguous automaton. The operators $\mathcal{U}$ and $\mathcal{R}$ are treated in similar fashion (see below).

Formula	[12] splits to		Tulip splits to	
$\varphi \vee \psi$	φ	ψ	φ	$\neg\varphi \wedge \psi$
$\varphi \,\mathcal{U}\, \psi$	ψ	$\varphi \wedge \bigcirc(\varphi \,\mathcal{U}\, \psi)$	ψ	$\neg\psi \wedge \varphi \wedge \bigcirc(\varphi \,\mathcal{U}\, \psi)$
$\varphi \,\mathcal{R}\, \psi$	$\varphi \wedge \psi$	$\psi \wedge \bigcirc(\varphi \,\mathcal{R}\, \psi)$	$\varphi \wedge \psi$	$\neg\varphi \wedge \psi \wedge \bigcirc(\varphi \,\mathcal{R}\, \psi)$

Example 1. Consider the incomplete Markov chain $\mathcal{M}$ with undefined transition probabilities, represented by variables x and y, shown in Figure 2. We optimise $\mathcal{M}$ with respect to the LTL formula $\varphi \stackrel{\text{def}}{=} \diamondsuit a \wedge \diamondsuit b$. The automaton $\mathcal{A}$ representing this formula and the product graph $\mathcal{G}_{\mathcal{M}\otimes\mathcal{A}}$ (with accepting vertices and transition probabilities) are also shown in Figure 2.

Considering $\mathcal{G}_{\mathcal{M}\otimes\mathcal{A}}$, the expected number of times for a run starting in vertex v_1 to visit vertices v_4 and v_5 is given by

$$\alpha(v_4) = \sum_{i=1}^{\infty} \left(\frac{x}{10}\right)^i = \frac{x}{10-x} \quad \text{and} \quad \alpha(v_5) = \sum_{i=1}^{\infty} \left(\frac{4y}{5}\right)^i = \frac{4y}{5-4y}$$

Furthermore, let $\beta(v)$ be the probability to reach an accepting vertex from v. Then we have $\beta(v_2) = \frac{y}{10-x}$ and $\beta(v_3) = \frac{4x}{5-4y}$.

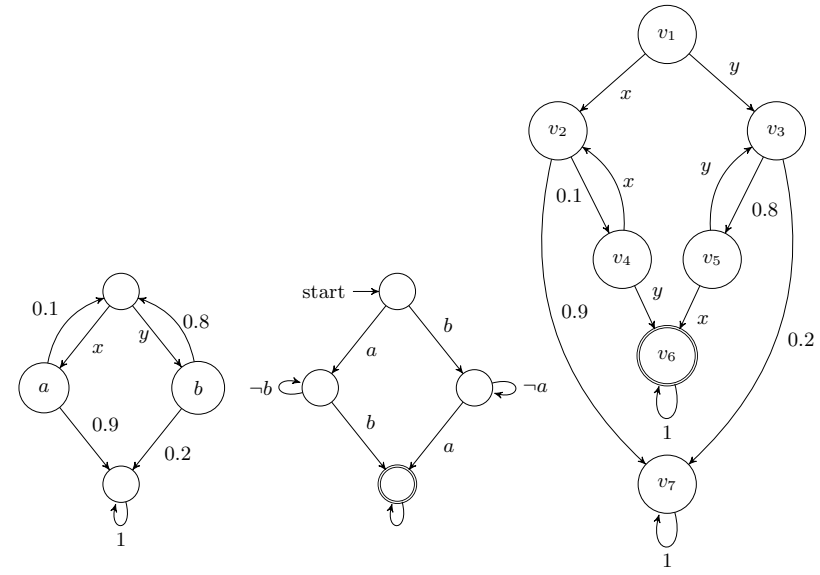
From (4) the expected number of times to take the x-labelled edge in $\mathcal{M}$ along a run that satisfies $\diamondsuit a \wedge \diamondsuit b$ is

$$\begin{aligned} f(x) &\stackrel{\text{def}}{=} \alpha(v_1)\cdot x \cdot \beta(v_2) + \alpha(v_4)\cdot x \cdot \beta(v_2) + \alpha(v_5)\cdot x \cdot \beta(v_6) \\ &= \frac{xy}{10-x} + \frac{x^2 y}{(10-x)^2} + \frac{4yx}{5-4y}\,. \end{aligned}$$

Likewise, the expected number of times to take the y-labelled edge in $\mathcal{M}$ along a run that satisfies $\diamondsuit a \wedge \diamondsuit b$ is

$$\begin{aligned} g(x) &\stackrel{\text{def}}{=} \alpha(v_1)\cdot y \cdot \beta(v_3) + \alpha(v_5)\cdot y \cdot \beta(v_3) + \alpha(v_4)\cdot y \cdot \beta(v_6) \\ &= \frac{4xy}{5-4y} + \frac{16xy^2}{(5-4y)^2} + \frac{xy}{10-x}\,. \end{aligned}$$

Now the sequence of transition values (x_n) defined by $x_{n+1} = \frac{f(x_n)}{f(x_n)+g(x_n)}$ converges to a limit ($\simeq 0.32$) that maximizes the probability of satisfying $\diamondsuit a \wedge \diamondsuit b$.



Interval Markov chain $\mathcal{M}$ Automaton $\mathcal{A}$ for $\Diamond a \wedge \Diamond b$ Product graph $\mathcal{G}_{\mathcal{M}\otimes\mathcal{A}}$

Fig. 2. Example

5 Implementation and Experiments

Our tool Tulip can be accessed at http://tulip.lenhardt.co.uk along with several examples. The tool inputs a labelled interval Markov chain along with properties specified either by LTL formulas or directly by unambiguous Büchi automata. It performs a specified number of iterations of the EM algorithm and outputs an approximation to the maximum probability with which the IMC satisfies the property, together with the values within the intervals for which the maximum is achieved.

LTL-to-Automaton Translation. We begin by comparing the performance of our translation component with other methods of generating automata from LTL. Our translation begins by pre-processing the formula using the simplifier of LTL2dstar [15], allowing us, for example, to notice that LTL formula $\neg(\Diamond\Diamond p_1 \leftrightarrow \Diamond p_1)$ is equivalent to *false*. The table below compares the unambiguous automata we construct with the experimental results of constructing non-deterministic automata reported in [12]. We took formulas from [12] covering a range of useful properties, such as fairness. We compare with [12] because the non-deterministic automata generated by [12] are already extremely small, as shown below. Our experiments suggest that the extra cost of producing unambiguous automata is usually very small, and so using Tulip we get nearly optimal unambiguous automata. This is encouraging given that unambiguous automata can be used directly for probabilistic model checking without determinization.

Moreover, assuming reasonable computational resources (1 GB of RAM and few seconds of CPU time), we were able to use Tulip to construct automata with up to $10,000$ nodes.

	[12]		Tulip	
Formula	Nodes	Edges	Nodes	Edges
$p_1 \,\mathcal{U}\, p_2$	3	4	3	5
$p_1 \,\mathcal{U}\, (p_2 \,\mathcal{U}\, p_3)$	4	6	4	9
$\neg(p_1 \,\mathcal{U}\, (p_2 \,\mathcal{U}\, p_3))$	7	15	5	12
$\Box\Diamond p_1 \rightarrow \Box\Diamond p_2$	9	15	8	18
$\Diamond p_1 \,\mathcal{U}\, \Box p_2$	8	15	6	16
$\Box(p_1 \,\mathcal{U}\, p_2)$	5	6	4	10
$\neg(\Diamond\Diamond p_1 \leftrightarrow \Diamond p_1)$	22	41	1	1

In general, non-deterministic automata can be exponentially more succinct than unambiguous automata. There are also cases, such as e.g. LTL formula $\Diamond(a \wedge \bigcirc^k a)$, when Tulip translates the formula into an automaton with number of states exponential in k. However, as our comparison with PRISM illustrates below, even in this case Tulip can still produce much smaller automata by avoiding the need to determinize.

Optimizations on Product Chains. Here we describe some of the optimizations that we use to reduce the state space of automata and the cross-product of automata and IMCs. We apply probabilistic bisimulation to the cross product, extending the usual notion to handle parameters. In a step of an iterative refinement algorithm for (standard) probabilistic bisimulation, one has to match the total mass of transitioning from a state u to some equivalence class E with the mass passing from a state v to E. In the case of our cross product machine, our transitions are labeled with parameters from the IMC, and our notion of matching is as a formal sum. We have found that the time spent performing bisimulation was more than compensated for by allowing faster iterations and reduced memory consumption of the EM algorithm,

Another opportunity to reduce the state space is to collapse vertices. First note that when we form a cross product between an IMC and an unambiguous automaton we can determine nodes that are "almost surely accepting" (i.e. starting at this vertex we will accept with probability one), just by checking the underlying structure of the graph. More generally, vertices can be grouped together if almost every accepting path that goes through one must go through another. For example, vertices in a bottom SCC can be collapsed into a single vertex, and linear subgraphs that do not contain accepting vertices can be collapsed.

Benchmarks. To test the effect of our automata translation techniques on the performance of LTL model checking, we consider a simple *Probabilistic Broadcast Protocol (PBP)* [11] by which nodes in a network propagate information. In this protocol when a node receives a message, it broadcasts the message to its neighbours with a certain probability and otherwise ignores the message. In either case the node then goes to sleep. We model a synchronous variant with

message collision: all sending and receiving is in rounds, and if a node is sent a message simultaneously from two neighbours it only receives noise. Tulip imports an existing Markov-chain model of the protocol from PRISM. There are no interval transitions in this model.

We model check the LTL property $\Diamond(a \wedge \bigcirc^k a)$ for various values of the parameter k, where a denotes the sending of a message to a given node in the network. The table below gives the outcome, showing how Tulip outperforms PRISM on this example. We attribute the latter to Tulip's use of unambiguous automata, whereas PRISM relies on a complex determinization construction. The Markov chain in this example is relatively small, so PRISM's symbolic model checking capability is not exploited.

k	2	3	4	5	6	7	8	9	10
Tulip	0.017	0.026	0.065	0.072	0.140	0.292	0.471	0.859	1.412
PRISM	0.015	0.023	0.040	0.111	0.369	0.864	1.820	6.465	30.101

Now we turn to the case of Interval Markov Chains, considering all stages of our algorithm. We evaluate the performance of Tulip using a single core of 1.7 Ghz Intel Core i5 CPU. The first column contains results for the interval Markov chain from Example 1. The second column contains results for model checking the *Bounded Retransmission Protocol (BRP)* [9]. The BRP splits a given file into N chunks and tries to send each of them at most MAX times, using two lossy channels for transmissions and acknowledgements. In contrast to prior modeling of this protocol (e.g. in PRISM), we do not model message losses by a fixed probability but by intervals representing a range estimate on their reliability. We set $N = 32$, $MAX = 3$ and model check the property that *the sender does not report a successful transmission.*

Interval Markov chain	Example 1		BRP	
LTL property	$\Diamond(a \wedge \bigcirc^8 b)$		$\Diamond a$	
	Nodes	Time(s)	Nodes	Time(s)
Initial automaton	1026	0.142	5	0.000
Automaton after bisimulation	513	0.035	3	0.002
Naive cross product	2052	0.004	5301	0.142
Product with reachable states only	122		1767	
Product after collapse	74	0.008	610	23.944
Product after bisimulation	72	0.009	544	2.139
One iteration of EM algorithm		0.003		0.934

Each iteration of our algorithm runs in cubic time, so the above techniques reducing the size of the product chain are worthwhile. For example, in our benchmarks it could be seen that an iteration on an example with over 500 nodes took less than a second. In the examples above and below, at most tens of iterations are sufficient to attain a precision up to five decimal places. For example, our algorithm stabilized to this level of accuracy in four iterations for the model from Figure 2 (a solution found to be the correct global maximum by hand analysis); we needed only one iteration for the BRP model.

Below we show the impact of our optimizations on additional examples which are described on our website. They cover a range of scenarios, including finding mixed strategies in some economic games and evaluating properties specifying competing goals.

Examples: *Rendezvous in the Park* (R), *Competing Goals* (G), *Modifying Dice* (D), *Predicting Football* (F), *Probabilistic Broadcast Protocol* (P).

	R	G	D	F	P
Size of interval Markov chain	5	4	7	22	79
Initial automaton size	6	10	10	82	162
Automaton after bisimulation	6	4	9	29	129
Naive cross product	30	20	63	638	10191
Product with reachable states only	20	11	21	171	599
Product after collapse	7	9	12	41	83
Product after bisimulation	6	8	6	15	18
Iterations for 5-decimal-digit precision	14	6	1	12	1
Start to end running time (in seconds)	0.013	0.007	0.010	0.024	0.140

The results support our observations above concerning the size of automata generated, the speed of a particular iteration, and the number of iterations required.

6 Conclusions

In this work we show that the IMC model has advantages in complexity of evaluation over general MDPs. This is reflected in our worst-case bounds, and also at the pragmatic level. We are able to avoid translation to deterministic automata, which is essential to MDP solving for LTL specifications, making do instead with unambiguous automata. We are also able to make use of methods for parameter training from other areas. In this paper we have focused on EM, but in future work we will look at adaptations of other training methods, such as gradient descent.

For specifications given by automata, our $\mathrm{NP}_{\mathbb{R}}$-completeness result shows that the complexity of IMC model-checking lies in PSPACE. Note that a PSPACE-hardness result would imply that satisfiability for the existential theory of the reals is PSPACE-hard, while the complexity of this theory has been open for quite some time. For LTL specifications, our results only isolate the complexity between PSPACE and EXPSPACE. We will look for tighter bounds in future work.

References

1. Naoki Abe and Manfred K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260, 1992.

2. Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.
3. Michael Ben-Or, Dexter Kozen, and John H. Reif. The complexity of elementary algebra and geometry. *JCSS*, 32(2):251–264, 1986.
4. Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag, 1997.
5. Allan Borodin, Stephen A. Cook, and Nicholas Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58(1-3):113–136, 1983.
6. John F. Canny. Some algebraic and geometric computations in pspace. In *STOC*, 1988.
7. Krishnendu Chatterjee, Koushik Sen, and Thomas A. Henzinger. Model-checking omega-regular properties of interval markov chains. In *FoSSaCS*, 2008.
8. J.-M. Couvreur, N. Saheb, and G. Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *LPAR*, 2003.
9. Pedro D'Argenio, Bertrand Jeannet, Henrik Jensen, and Kim Larsen. Reachability analysis of probabilistic systems by successive refinements. In *PAPM/PROBMIV*, 2001.
10. Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. Decision problems for interval markov chains. In *LATA*, 2011.
11. A. Fehnker and P. Gao. Formal verification and simulation for performance analysis for probabilistic broadcast protocols. In *ADHOC-NOW*, 2006.
12. Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification Testing and Verification*, 1995.
13. Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer, 1993.
14. Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS*, 1991.
15. Joachim Klein and Christel Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2):182–195, 2006.
16. Christos Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.
17. Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.