Probabilistic verification and approximation

Richard Lassaigne[1] and Sylvain Peyronnet[1,2]

[1] Equipe de Logique,
UMR 7056 CNRS,
University Paris VII
lassaign@logique.jussieu.fr
[2] EPITA Research and Development Laboratory (LRDE)
syp@lrde.epita.fr

Abstract. Model checking is an algorithmic method allowing to automatically verify if a system which is represented as a Kripke model satisfies a given specification. Specifications are usually expressed by formulas of temporal logic. The first objective of this paper is to give an overview of methods able to verify probabilistic systems. Models of such systems are labelled discrete time Markov chains and specifications are expressed in extensions of temporal logic by probabilistic operators. The second objective is to focus on complexity of these methods and to answer the question: **can probabilistic verification be efficiently approximated?** In general, the answer is negative. However, in many applications, the specification formulas can be expressed in some positive fragment of linear time temporal logic. One will show how some simple randomized approximation algorithms can improve the efficiency of verification for probabilities acting on useful classes of formulas.

1 Introduction

Model checking has been successfully used to verify if the behavior of concurrent and reactive systems satisfy some correctness properties, which are usually expressed in temporal logic. While many important system properties can be studied in this framework, others such as reliability and performance, require instead a probabilistic characterisation of the system.
In the case of classical model checking, time complexity is linear in the size of the model and in the size of formula for CTL branching time temporal logic [7], and exponential in the size of formula for LTL linear time temporal logic [23]. In many practical cases the representation of systems by Kripke models leads to a "state explosion phenomenon" and symbolic representation methods of the transition system and of the formula have been introduced to overcome this problem. Ordered Binary Decision Diagrams (OBDDs)[5], or equivalently read-once branching programs with an ordering restriction on the variables, provide in many practical cases a compact representation of the transition relation and of the formula to be checked. However, in some cases, such as integer multiplication, the OBDD size is exponential [6].

For probabilistic verification, time complexity is in general polynomial in the size of the model and polynomial or exponential in the size of the formula [8]. For the same reason as in classical model checking, symbolic representation methods have been generalized in the probabilistic framework. However, in spite of using these techniques, for many examples the verification is limited to small values of the characteristic parameters of the model. Thus it seems natural to wonder whether some approximation algorithms could be useful for probabilistic verification. Unfortunately, there are serious complexity reasons to think that this is not the case in general. Nevertheless, there exists some feasible algorithms to approximate model checking of various classes of LTL formulas against probabilistic transition systems. These randomized algorithms allows to approximate, with very high confidence, satisfaction probability of such formulas and to decide if this probability is greater than a given threshold. The main advantage is to eliminate space complexity by using a Monte-Carlo method and efficiently bounding sample size. Moreover this approach is highly parallelizable and can be integrated in a classical probabilistic model checker.
Our main results are:

- $\sharp P$-completeness of counting paths satisfying LTL formulas;
- non-approximability of computing associated probabilities;
- a randomized approximation scheme for computing probabilities of the form $p = Prob[\psi]$ for formulas ψ in some positive fragment of LTL and $p \in]0, 1[$.

Section 2 is a review of the main classical results in probabilistic verification and of the existing probabilistic model checkers. In section 3, we describe the linear time framework for probabilistic verification and we show the hardness of approximating probabilities for LTL formulas. In section 4, we give a randomized approximation algorithm to compute such probabilities when we restrict the specification language to some positive fragment of LTL and probabilities to the open interval $]0, 1[$.

2 Classical probabilistic verification

2.1 Qualitative verification

The first application of verification methods to probabilistic systems consisted in checking if temporal properties are satisfied with probability 1 by systems modeled either as finite discrete time Markov chains or as Markov models enriched with nondeterministic behavior. In the following, the former systems will be denoted by probabilistic sytems and the latter by concurrent probabilistic sytems.
In [28], Vardi presented the first method to verify if a linear time temporal property is satisfied by almost all computations of a concurrent probabilistic system. This automata-theoretic method is expensive, as it is doubly exponential in the size of the formula.
The complexity of this work was later adressed by Courcoubetis and Yannakakis

[8]. A new model checking method for probabilistic systems was introduced, the complexity of which was proved polynomial in the size of the system and exponential in the size of the formula. For concurrent probabilistic systems they presented an automata-theoretic approach which improved on the Vardi's method by a single exponential in the size of the formula.

2.2 Quantitative verification

The Courcoubetis and Yannakakis's method [8] allows to compute the probability that a probabilistic system satisfies some given linear time temporal formula. A temporal logic for the specification of quantitative properties, which refer to a bound of the probability of satisfaction of a formula, was given by Hansson and Jonsson [14]. The authors introduced the logic $PCTL$, which is an extension of branching time temporal logic CTL with some probabilistic quantifiers. A model checking algorithm was also presented: the computation of probabilities for formulas involving probabilistic quantification is performed by solving a linear system of equations.
A model checking method for probabilistic concurrent systems against $PCTL$ and $PCTL^*$ (the standard extension of $PCTL$) properties is given by Bianco and de Alfaro [3]. Probabilities are computed by solving an optimisation problem over system of linear inequalities, rather than linear equations as in [14]. The algorithm for the verification of $PCTL^*$ is obtained by a reduction to the $PCTL$ model checking problem using a transformation of both the formula and the probabilistic concurrent system. Model checking of $PCTL$ formulas is shown to be polynomial in the size of the system and linear in the size of the formula, while $PCTL^*$ verification is polynomial in the size of the system and doubly exponential in the size of the formula.

We mention model checking tools that were designed for the verification of quantitative properties. In general, these tools use extensions of OBDDs called Multi-Terminal Binary Decision Diagrams (MTBDDs) to represent Markov transition matrices. ProbVerus [13] uses $PCTL$ model checking and symbolic techniques to verify $PCTL$ formulas on fully probabilistic systems. PRISM [10] is a probabilistic model checker which allows to check $PCTL$ formulas on fully or concurrent probabilistic sytems. The *Erlangen-Twente Markov Chain Checker* [15] ($E \vdash MC^2$) supports model checking of continuous-time Markov chains against specifications expressed in continuous-time stochastic logic (CSL). Numerous classical protocols represented as probabilistic or concurrent probabilistic systems have been successfully verified by PRISM. But experimental results are often limited by the exponential blow up of space needed to represent the transition matrice and to solve linear systems of equations or inequations.
In this context, it is natural to ask the question: **can probabilistic verification be efficiently approximated?** In the following, we study some possible answers for fully probabilistic systems and linear time temporal logic.

3 Probabilistic verification and approximation

In this section, we present the classical logical framework which allows to express quantitative properties of probabilistic systems, the results of Courcoubetis and Yannakakis and the hardness of approximating such probabilities.

3.1 Probabilistic verification

A Discrete Time Markov Chain (DTMC) is a pair $\mathcal{M} = (S, P)$ where S is a finite or countable set of states and $P : S \times S \to [0, 1]$ is a transition probability function, i.e. for all $s \in S$, $\sum_{t \in S} P(s, t) = 1$. If S is finite, we can consider P as a transition matrix.

Definition 1. *A probabilistic transition system (PTS) is a structure $\mathcal{M} = (S, P, s_0, L)$ given by a Discrete Time Markov chain (S, P) with an initial state s_0 and a function $L : S \to \mathcal{P}(AP)$ labelling each state with a set of atomic propositions in AP.*

A path σ is a finite or infinite sequence of states $(s_0, s_1, \dots, s_i, \dots)$ such that $P(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We denote by $Path(s)$ the set of paths whose first state is s.
In linear-time logic LTL, formulas are composed from the set of atomic propositionsby using the boolean connectives and the temporal operators **X** (*next*) and **U** (*until*). The usual operators **F** (*eventually*) and **G** (*always*) can also be defined. LTL formulas are interpreted over paths of a transition system $\mathcal{M}$. A path σ is a finite or countable sequence of states $(s_0, s_1, \dots, s_i, \dots)$ such that $(s_i, s_{i+1}) \in R$ for all $i \geq 0$. We note also $\sigma(i)$ the $(i + 1)$th state of the path σ and σ^i the path $(\sigma(i), \sigma(i + 1), \dots)$.

Definition 2. *The interpretation of LTL formulas over paths is defined by:*

- $\mathcal{M}, \sigma \models p$ *iff* $p \in L(\sigma(0))$.
- $\mathcal{M}, \sigma \models \neg\phi$ *iff* $\mathcal{M}, \sigma \not\models \phi$.
- $\mathcal{M}, \sigma \models \phi \wedge \psi$ *iff* $\mathcal{M}, \sigma \models \phi$ *and* $\mathcal{M}, \sigma \models \psi$.
- $\mathcal{M}, \sigma \models \mathbf{X}\phi$ *iff* $\mathcal{M}, \sigma^1 \models \phi$.
- $\mathcal{M}, \sigma \models \phi \mathbf{U} \psi$ *iff there exists* $j \geq 0$ *s.t.* $\mathcal{M}, \sigma^j \models \psi$ *and for all* $i < j$ $\mathcal{M}, \sigma^i \models \phi$.

We say that an LTL formula ϕ is universally valid in $\mathcal{M}$, which we write $\mathcal{M} \models \forall\phi$, iff for all paths $\sigma \in Path(s_0)$, $\mathcal{M}, \sigma \models \phi$. An LTL formula ϕ is existentially valid in $\mathcal{M}$, which we write $\mathcal{M} \models \mathbf{E}\phi$, iff there a paths $\sigma \in Path(s_0)$ such that $\mathcal{M}, \sigma \models \phi$.
For each structure $\mathcal{M}$ and state s, it is possible to define a probability measure $Prob$ on the set $Path(s)$:
for any finite path $\pi = (s_0, s_1, \dots, s_n)$, the measure is defined by
$Prob(\{\sigma/\sigma \text{ is a path and } (s_0, s_1, \dots, s_n) \text{ is a prefix of } \sigma\}) = \prod_{i=1}^{n} P(s_{i-1}, s_i)$.
This measure can be extended uniquely to the Borel family of sets generated by the sets $\{\sigma/\pi \text{ is a prefix of } \sigma\}$ where π is a finite path.

In [28], it is shown that for any LTL formula ϕ, probabilistic transition system $\mathcal{M}$ and state s, the set of paths $\{\sigma/\sigma(0) = s \text{ and } \mathcal{M}, \sigma \models \phi\}$ is measurable. We denote by $Prob[\phi]$ the measure of this set. We say that the probabilistic transition sytem $\mathcal{M}$ satisfy the formula ϕ if $Prob[\phi] = 1$, i.e. if almost all paths in $\mathcal{M}$, whose origin is the initial state, satisfy ϕ.

Theorem 1. *([8])*
The satisfaction of a LTL formula ϕ by a probabilistic transition sytem $\mathcal{M}$ can be decided in time linear in size of $\mathcal{M}$ and exponential in size of ϕ, or in space polylogarithmic in size of $\mathcal{M}$ and polynomial in size of ϕ.
The probability $Prob[\phi]$ can be computed in time polynomial in size of $\mathcal{M}$ and exponential in size of ϕ.

The first part of the theorem is about qualitative properties, the second one concerns quantitative properties. In the following, we study quantitative properties and show,in the next subsection, that the problem of computing the number of truth assignments satisfying a propositional formula in conjunctive normal form ($\sharp SAT$) can be reduced to the problem of counting finite paths satisfying LTL formulas. Therefore it is unlikely to obtain an approximation algorithm for computing $Prob[\phi]$ in general.

3.2 Counting problems and approximation

Intuitively, the class $\sharp P$ captures the problems of counting the numbers of solutions to NP problems. The counting versions of all known NP-complete problems are $\sharp P$-complete. The well adapted notion of reduction is parsimonious reduction: it is a polynomial time reduction from the first problem to the second one, recovering via some oracle, the number of solutions for the first problem from the number of solutions for the second one.
No deterministic approximation algorithms are known for $\sharp P$-complete problems. However, randomized versions of such approximation algorithms exist for problems such as counting the number of valuations satisfying a propositional disjunctive normal form formula ($\sharp DNF$) [21] or network reliability problem [19]. We introduce the notion of polynomial randomized approximation scheme which is due to Karp and Luby [20]. Consider a counting problem and let $\#(x)$ the number of distinct solutions for an instance x of this problem. We note $|x|$ the size of this instance.

Definition 3. *A randomized approximation scheme (RAS) for a counting problem is a randomized algorithm $\mathcal{A}$ that takes an input x, a real number $\epsilon > 0$ and produces a value $A(x, \epsilon)$ such that for any x, $\epsilon > 0$, and $\delta > 0$:*

$$Pr\big(|A(x,\epsilon) - \#(x)| \leq \epsilon.\#(x)\big) \;\geq\; 1 - \delta.$$

A randomized approximation scheme is said to be fully polynomial (FPRAS) [20] if its running time is polynomially bounded in $|x|$, $\frac{1}{\epsilon}$ and $\log(\frac{1}{\delta})$.

The probability Pr is taken over the random choices of the algorithm. We call ϵ the *error parameter* and δ the *confidence parameter*. We remark that the error parameter is a multiplicative parameter taking into account the size of $\#(x)$.
A probability problem is defined by giving as input a succinct representation of a probabilistic system, a property x and as output the probability measure $\mu(x)$ of the measurable set of execution paths satisfying this property.

Definition 4. *A randomized approximation scheme for a probability problem is a randomized algorithm $\mathcal{A}$ that takes an input x and a real number $\epsilon > 0$ and produces a value $A(x, \epsilon)$ such that for any x, $\epsilon > 0$, and $\delta > 0$:*

$$Pr\big(|A(x,\epsilon) - \mu(x)| < \epsilon.\mu(x)\big) \;\geq\; 1 - \delta.$$

If the running time of $\mathcal{A}$ is polynomial in $|x|$, $\frac{1}{\epsilon}$ and $\log(\frac{1}{\delta})$, $\mathcal{A}$ is said to be fully polynomial.

We consider the fragment $L(\mathbf{F})$ of LTL in which $\mathbf{F}$ is the only temporal operator.

Theorem 2. *The problem of counting finite paths satisfying $L(\mathbf{F})$ formulas is $\sharp P$-complete.*

The following result is due to Clarke and Sistla [27]: the problem of deciding the existence of some path satisfying a $L(\mathbf{F})$ formula in a transition system is NP-complete. Their proof uses a polynomial time reduction of SAT to the problem of deciding satisfaction for formulas of $L(\mathbf{F})$. From this reduction, we can obtain a parcimonious reduction between $\sharp SAT$ formula and counting finite paths satisfying the associated $L(\mathbf{F})$ formula.
Some consequence of this theorem is the $\sharp P$-hardness of computing probabilities of satisfaction for general LTL formulas. We remark that if there was a FPRAS for approximating $Prob[\phi]$ for LTL formula ϕ, we could efficiently approximate $\sharp SAT$. We recall the randomized complexity class BPP which corresponds to the useful class of two-sided error randomized algorithms. Let Σ be some finite alphabet and Σ^* be the set of strings over Σ.

Definition 5. *The class BPP, for Bounded-error Probabilistic Polynomial time, consists of all languages L that have a randomized polynomial time algorithm $\mathcal{A}$ such that for any input $x \in \Sigma^*$,*

- *if $x \in L$ then $Pr\big(\mathcal{A}(x)\ accepts\big) \geq \frac{3}{4}$,*
- *if $x \notin L$ then $Pr\big(\mathcal{A}(x)\ accepts\big) \leq \frac{1}{4}$*

A polynomial randomized approximation scheme for $\sharp SAT$ could be used to distinguish, for input x, between the case $\sharp(x) = 0$ and the case $\sharp(x) > 0$, thereby implying a randomized polynomial time algorithm for the decision version SAT.

Corollary 1. *There is no fully polynomial randomized approximation scheme for the problem of computing $Prob[\phi]$ for LTL formula ϕ, unless $BPP = NP$.*

As a consequence of a result of Jerrum and Sinclair [18], $\sharp P$-complete problems either admit an FPRAS or are not approximable at all. Therefore there are no deterministic polynomial time approximation algorithms neither for $\sharp SAT$ nor for computing $Prob[\phi]$ for the $L(\mathbf{F})$ fragment of LTL.

4 Approximate probabilistic model checking

A first natural approach issue from bounded model checking, that is a method designed by [4] to check properties expressed by LTL formulas against deterministic transition systems.

4.1 Probabilistic bounded model checking

Biere, Cimatti, Clarke and Zhu [4] present a symbolic model checking technique using SAT procedures instead of BDDs. They introduce bounded model checking (BMC), where the bound corresponds to the maximal length of a possible counterexample. First, they give a correspondance between BMC and classical model checking. Then they show how to reduce BMC to propositional satisfiability in polynomial time.
The bounded model checking procedure works as follows. Given a transition system $\mathcal{M}$, an LTL formula ϕ and a bound $k \in \mathbb{N}$, they construct a propositional formula which is satisfiable if and only if there exists a path of length k which is a counterexample to the specification expressed by ϕ. This procedure is well adapted to finding a counterexample, if it exists, by incrementing the bound k.
Let us review more precisely what BMC is. Given a transition system $\mathcal{M}$, an LTL formula ϕ and a bound k, if we want to verify $\mathcal{M} \models \forall\phi$, we consider an LTL formula ψ which is in positive normal form and is equivalent to $\neg\phi$. Then the translation of the formula ψ to a propositional formula is given in two parts: the first component $[\![\mathcal{M}]\!]_k$ means for a sequence $(s_0, s_1, \ldots, s_k)$ to be a path σ in $\mathcal{M}$ and the second component $[\![\psi]\!]_k$ forces σ to satisfy ψ. The following theorem summarizes the results of [4] for bounded model checking of LTL formulas.

Theorem 3. *[4]*
Let ψ be an LTL formula and $\mathcal{M}$ be a transition system. Then $\mathcal{M} \models \mathrm{E}\psi$ if and only if there exists $k = O(|\mathcal{M}.2^{|\psi|})$ such that $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$ has a satisfying assignment.

To check the initial property ϕ, one should look for the existence of a counterexample to ψ for a given k, i.e. a satisfying assignment of $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$. If one does not find such a counterexample for $k \leq |S| \times 2^{|\psi|}$, then the initial property is true. We cannot hope to find a polynomial bound on k with respect to the size of S and ψ, since the model checking problem for LTL is PSPACE-complete (see [27]) and one could have a polynomial reduction to propositional satisfiability.
For practical verification of probabilistic protocols, quantitative properties have often the following form: $Prob[\psi] \geq b$ for a treshold value $b \in [0, 1]$. We try to check $Prob[\psi] \geq b$ by considering $Prob_k[\psi] \geq b$, i.e. the probability measure restricted to the probabilistic space defined by execution paths of length k. Following the BMC approach, we can associate to a formula ψ and length k the propositional formula $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$ in such a way that a path of length k satisfying ψ corresponds to an assigment satisfying $[\![\mathcal{M}]\!]_k \wedge [\![\psi]\!]_k$. Determining $Prob_k[\psi]$ is thus reduced to a counting version of SAT. Unfortunately, not only no efficient algorithms are known to solve such counting problems, but they are believed to be strongly non-approximable by deterministic algorithms [24].

4.2 A positive fragment of LTL

For many natural properties, satisfaction on an execution path of length k implies satisfaction by any extension of this path. Such properties are called monotone. We consider a subset of LTL formulas which allows to express only monotone properties and for which we can deduce corresponding bounds on satisfaction probabilities.

Definition 6. *The* essentially positive fragment *(EPF) of LTL is the set of formulas constructed from atomic formulas (p) their negations ($\neg p$), closed under $\vee$, $\wedge$ and the temporal operators X, U.*

These formulas include nested compositions of U but do not allow for negations in front. Nevertheless, this fragment can express various classical properties of protocols as accessibility, livelock freeness and convergence. If ϕ is a formula of the EPF fragment, we can use a BMC-like framework to verify whether ϕ is true on a path σ of length k. The monotonicity of the property defined by an EPF formula gives the following result.

Proposition 1. *Let ψ be an LTL formula of the essentially positive fragment and $\mathcal{M}$ be a probabilistic transition system. Then the sequence $(Prob_k[\psi])_{k\in\mathbb{N}}$ converges to $Prob[\psi]$.*

A first idea is to approximate $Prob_k[\psi]$ and to use a fixed point algorithm to obtain an approximation of $Prob[\psi]$. This approximation problem is believed to be intractable for deterministic algorithms. In the next section, we give a randomized approximation algorithm whose running time is polynomial in the size of a succinct representation of the system and of the formula.

4.3 Randomized approximation scheme with additive error

We show that we can approximate the satisfaction probability of an EPF formula with a simple randomized algorithm. As in many applications randomized approximation with additive error is sufficient and gives simple algorithms, we first explain how to design it. Then we will use the estimator theorem [21] and an optimal approximation algorithm [9] in order to obtain randomized approximation scheme with multiplicative error parameter, according to definition 4.
We generate random paths in the probabilistic space underlying the Kripke structure of depth k and compute a random variable A which additively approximates $Prob_k[\psi]$. Our approximation will be correct with confidence $(1-\delta)$ after a polynomial number of samples in $\frac{1}{\epsilon}, \log\frac{1}{\delta}$. This result is obtained by using Chernoff bounds [25] on the tail of the distribution of a sum of independent random variables.
The main advantage of the method is that we can proceed with just a succinct representation of the transition system, that is a succinct description in an input language, for example Reactive Modules [2]. Thus eliminating the space complexity problem.

Definition 7. *A succinct representation, or diagram, of a PTS $\mathcal{M} = (S, P, s_0, L)$ is a representation of the PTS, that allows to generate algorithmically, for any state s, the set of states t such that $P(s,t) > 0$.*

The size of such a succinct representation is subtantially lower than the size of the corresponding PTS. Typically, for Reactive Modules, the size of the diagram is polylogarithmic in the size of the PTS.
The following function **Random Path** uses this succinct representation to generate a random path of length k and to check the formula ψ:

Random Path
Input: $diagram_{\mathcal{M}}, k, \psi$
Output: samples a path π of length k and check formula ψ on π
1. Generate a random path π of length k (with the diagram)
2. If ψ is true on π then return 1 else 0

Consider now the random sampling algorithm $\mathcal{GAA}$ designed for the approximate computation of $Prob_k[\psi]$:

Generic approximation algorithm $\mathcal{GAA}$
Input: $diagram_{\mathcal{M}}, k, \psi, \epsilon, \delta$
Output: approximation of $Prob_k[\psi]$
$N := 4\ln(\frac{2}{\delta})/\epsilon^2$
$A := 0$
For $i = 1$ to N do $A := A +$ **Random Path**$(diagram_{\mathcal{M}}, k, \psi)$
Return A/N

Theorem 4. *The generic approximation algorithm $\mathcal{GAA}$ is a fully polynomial randomized approximation scheme (with additive error parameter) for computing $p = Prob_k[\psi]$ whenever ψ is in the EPF fragment of LTL and $p \in]0,1[$.*

4.4 Randomized approximation scheme with multiplicative error

We use a generalization of the zero-one estimator theorem [21] to estimate the expectation μ of a random variable X distributed in the interval $[0,1]$. The generalized zero-one estimator theorem [9] proves that if $X_1, X_2, \ldots, X_N$ are random variables independent and identically distributed according to X, $S = \sum_{i=1}^{N} X_i$, $\epsilon < 1$, and $N = 4(e-2).\ln(\frac{2}{\delta}).\rho/(\epsilon.\mu)^2$, then S/N is an (ϵ, δ)-approximation of μ, i.e.:

$$Pr\big(\mu(1-\epsilon) \leq S/N \leq \mu(1+\epsilon)\big) \geq 1 - \delta$$

where $\rho = max(\sigma^2, \epsilon\mu)$ is a parameter used to optimize the number N of experiments and σ^2 denotes the variance of X.
In [9], an optimal approximation algorithm,running in three steps, is described.

- using a stopping rule, the first step outputs an (ϵ, δ)-approximation $\hat{\mu}$ of μ after expected number of experiments proportional to Γ/μ where $\Gamma = 4(e-2).\ln(\frac{2}{\delta})/\epsilon^2$;

– the second step uses the value of $\hat{\mu}$ to set the number of experiments in order to produce an estimate $\hat{\rho}$ that is within a constant factor of ρ with probability at least $(1 - \delta)$;
– the third step uses the values of $\hat{\mu}$ and $\hat{\rho}$ to set the number of experiments and runs the experiments to produce an (ϵ, δ)-approximation of μ.

We obtain a randomized approximation scheme with multiplicative error by applying the optimal approximation algorithm $\mathcal{OAA}$ with input parameters ϵ, δ and the sample given by the function **Random Path** on a succinct representation of $\mathcal{M}$, the parameter k and the formula ψ.

Theorem 5. *The optimal approximation algorithm $\mathcal{OAA}$ is a fully polynomial randomized approximation scheme (with multiplicative error parameter) for computing $p = Prob_k[\psi]$ whenever ψ is in the EPF fragment of LTL and $p \in]0, 1[$.*

Thus a randomized approximation of $Prob[\psi]$ can be computed by an iterating fixed point algorithm with the following stopping condition: $(Prob_{k+1}[\psi] - Prob_k[\psi]) < \epsilon/2$.

Corollary 2. *The fixed point algorithm defined by iterating the optimal approximation algorithm $\mathcal{OAA}$ is a randomized approximation scheme for the probability problem $p = Prob[\psi]$ whenever ψ is in the EPF fragment of LTL and $p \in]0, 1[$.*

4.5 APMC: an implementation

In 2003, we start the design, together with Thomas Hérault (University of Paris XI), of a tool that implements the approximation method, with additive error, described in this paper. This tool [16], called APMC for Approximate Probabilistic Model Checker, was freely available [1] under GPL (Gnu Public License) and is under permanent development. APMC is now a probabilistic distributed model checker that uses a client/server computation model in order to speed up the verification process by distributing the **Random Path** function on a cluster of workstations (extensive tests with hundreds machines were done). The tool is easy to use since it is provided with a graphical user interface to enter the model, formula and the approximation parameters.
Since 2003, numerous experiments were done, such as the verification of various probabilistic distributed algorithms (mutual exclusion, dining philosophers, leader election...) and of the IEEE 802.3 CSMA/CD protocol (part of the ethernet protocol) [11]. We also released the core computation engine of APMC into a self-sufficient library, which is now fully integrated into the state-of-the-art probabilistic model checker PRISM [10].

5 Related work

Similar sampling methods have been used for statistical model checking. In [30], a procedure is described for verifying properties of discrete event systems based

on Monte-Carlo simulation and statistical hypothesis testing. In [26], a statistical method is proposed to model checking of black-box probabilistic systems against specifications given in a sublogic of continuous stochastic logic (CSL). These approaches differ strongly from ours by using statistical hypothesis testing instead of randomized approximation schemes.
Recently, in [12], a randomized algorithm for probabilistic model checking of safety properties expressed as LTL formulas was given. This approximation method uses the optimal approximation algorithm of [9] and is complementary of our since safety properties are equivalent to the negation of properties expressed by the essentialy positive fragment of LTL.

6 Conclusion

In this paper, we first adressed the general problem of approximating the probabilistic verification of any linear time temporal formula against probabilistic systems. We showed that, even for a simple fragment of LTL, satisfaction probabilities of such formulas are non-approximable unless some unlikely complexity conjecture holds. Nevertheless, we presented a randomized approximation scheme for the quantitative verification of positive LTL formulas, using an optimal approximation algorithm [9]. We started the design of such methods in 2002 [22], and to our knowledge, it was the first time that such randomized approximation methods were used for probabilistic verification. The main advantage of this approximation method is to eliminate the space complexity problem due to the state explosion phenomenon that occurs in the representation of protocols as probabilistic transition systems.

Acknowledgements. We would like to thank Thomas Hérault for his strong participation in the development of APMC, and Radu Grosu for his private communication.

References

1. APMC Website. http://apmc.berbiqui.org
2. R. Alur and T.A. Henzinger. Reactive modules. In *Proc. of the 11th IEEE Symposium on Logic in Computer Science*, pp. 207-218, 1996.
3. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. *Proc. FST&TCS*, LNCS, 1026:499-513, 1995.
4. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDD's. In *Proc. of 5th TACAS* , LNCS, 1573:193–207, 1999.
5. R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.
6. R.E. Bryant. On the complexity of vlsi implementations and graph representations of boolean functions with application to integer multiplication. *IEEE Transactions on Computers*, 40(2):205–213, 1991.
7. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

8. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
9. P. Dagum, R. Karp, M. Luby and S. Ross. An optimal algorithm for Monte-Carlo estimation. *SIAM journal of computing*, 29(5):1484–1496, 2000.
10. L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic model checking of concurrent probabilistic processes using MTBDDs and the kronecker representation. In *Proc. of Int. TACAS*, LNCS, 1785, 2000.
11. M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. In *Proc. of the 4th AVoCS*, ENTCS, 2004.
12. R. Grosu. *Private communication.* 2005.
13. V. Hartonas-Garmhausen, S. Campos, and E. Clarke. Probverus: Probabilistic symbolic model checking. In *5th International AMAST Workshop, ARTS'99*, LNCS 1601, 1999.
14. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.
15. H. Hermans, J.P. Katoen, J. Meyer-Kayser, and M. Siegle. A Markov chain model checker. In *Proc. of Int. TACAS*, LNCS 1785, 2000.
16. T. Herault, R. Lassaigne, F. Magniette and S. Peyronnet. Approximate Probabilistic Model Checking. In *Proceedings of Fifth International VMCAI'04*, LNCS, 2937:73–84, 2004.
17. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13-30, 1963.
18. M.R. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal of Computing*, 18:1149–1178, 1989.
19. D.R. Karger. A randomized fully polynomial time approximation scheme for the all terminal network reliability problem. *SIAM journal on computing*, 29:492–514, 1999.
20. R.M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. *In Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science*, 56–64, 1983.
21. R.M. Karp, M. Luby and N. Madras. Monte-Carlo algorithms for enumeration and reliability problems. *Journal of Algorithms*, 10:429–448, 1989.
22. R. Lassaigne and S. Peyronnet. Approximate Verification of Probabilistic Systems. In *Proc. of the 2nd joint PAPM-PROBMIV*, LNCS, 2399:213–214, 2002.
23. O. Lichtenstein and A. Pnueli. *Checking that finite state concurrent programs satisfy their linear specification.* Proceedings of the 12th POPL, ACM CS Press, p.97-107, 1985.
24. C.H. Papadimitriou *Computational Complexity* Addison Wesley, 1994.
25. A. Rényi. *Probability Theory.* North-Holland, Amsterdam, 1970.
26. K. Sen, M. Vishanathan, and G. Agha. Statistical model checking of black-box probabilistic systems . In *Proc. of the 16th Int. Conf. Computer Aided Verification*, LNCS, 3114:202–215, 2004.
27. A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
28. M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. *Proc. 26th FOCS*, pp. 327–338, 1985.
29. V.V. Vazirani. *Approximation Algorithms.* Springer-Verlag, 2001.
30. H. L. S. Younes and R. G. Simmons. Probabilistic Verication of Discrete Event Systems using Acceptance Sampling. *In Proc. of the 14th International Conference on Computer Aided Verification*, LNCS, 2404:223–235. 2002.