This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

Asymptotic Perturbation Bounds for Probabilistic Model Checking with Empirically Determined Probability Parameters

Guoxin Su, Yuan Feng, Taolue Chen, and David S. Rosenblum, *Fellow, IEEE*

Abstract—Probabilistic model checking is a verification technique that has been the focus of intensive research for over a decade. One important issue with probabilistic model checking, which is crucial for its practical significance but is overlooked by the state-of-the-art largely, is the potential discrepancy between a stochastic model and the real-world system it represents when the model is built from statistical data. In the worst case, a tiny but nontrivial change to some model quantities might lead to misleading or even invalid verification results. To address this issue, in this paper, we present a mathematical characterization of the consequences of model perturbations on the verification distance. The formal model that we adopt is a parametric variant of discrete-time Markov chains equipped with a vector norm to measure the perturbation. Our main technical contributions include a closed-form formulation of *asymptotic perturbation bounds*, and computational methods for two arguably most useful forms of those bounds, namely *linear bounds* and *quadratic bounds*. We focus on verification of reachability properties but also address automata-based verification of omega-regular properties. We present the results of a selection of case studies that demonstrate that asymptotic perturbation bounds can accurately estimate the maximum variations of the verification results induced by the model perturbations.

Index Terms—Asymptotic perturbation bound, Discrete-time Markov chain, Numerical iteration, Optimization, Parametric Markov chain, Perturbation analysis, Probabilistic model checking, Quadratic programming

1 Introduction

Probabilistic model checking is a system verification technique that has matured over the past two decades and has been applied in software engineering, such as verification of non-functional requirements for complex software systems [10]. A common scenario of probabilistic model checking is to verify a system model, such as a Discrete-Time Markov Chain (DTMC) [41], Markov Decision Process (MDP) [44] and Continuous-Time Markov Chain (CTMC) [4], against a temporal property, such as a formula in the Linear Temporal Logic (LTL) [53] or Probabilistic Computation Tree Logic (PCTL) [30], and to return either a qualitative answer (namely a yes/no answer) or a quantitative answer (namely a probability). PRISM [34] is one of the most widely used probabilistic model checking tools.

Many case studies reported for probabilistic model checking, including those performed with PRISM, involve stochastic models embodying *theoretically defined* distributions, such as the use of a fair coin toss to introduce randomization into an algorithm, or the uniform probability distribution of randomly chosen IP addresses in the Zeroconf protocol. However, real-world systems contain probability parameters that are *empirically determined*, such as the failure rate of a system component. Whether the verification reflects the true quantitative property of the system underlying the stochastic model is dependent on whether the model is a faithful abstraction of the system. In the stochastic model construction, measurements or experiments are employed to determine the transition probabilities (for discrete-time systems) or transition rates (for continuous-time systems). On the one hand, those statistical quantities are affected by the measurement or experimental environment. For example, the rate of losing a message in a communication protocol implemented in a physical network is affected by network load, electrical or wireless noise, etc. On the other hand, the stochastic nature of the system itself may vary over time. For example, the reliability of a hardware component decreases with the age of the component.

In both of the above situations, usually the model builder is able to improve the precision of the empirical parameters with various measures, e.g., by increasing the sample size or reducing the environmental disturbance. However, it is important to consider the possible consequence of some *perturbation* occurring in the parameters on the verification of the model. In the worst case, a tiny but non-trivial change to some quantities in the model might lead to a misleading or even invalid verification result.

A straightforward method to address the above problem is to perform multiple point-wise model checking which is supported by e.g., the tool PRISM: We modify the values of the quantities in the model, run the model checker for each choice of values, and then compare the resulting set of outcomes. Such a simplified method only partially reveals the dependence of verification on model perturbations. A

- *G. Su and D. S. Rosenblum are with Department of Computer Science, School of Computing, National University of Singapore. E-mail: {sugx, david}@comp.nus.edu.sg*
- *Y. Feng is with Centre for Quantum Computation and Intelligent Systems, University of Technology Sydney, and AMSS-UTS Joint Research Laboratory for Quantum Computation, Chinese Academy of Sciences. E-mail: yuan.feng@uts.edu.au*
- *T. Chen is with Department of Computer Science, Middlesex University London. E-mail: t.l.chen@mdx.ac.uk*

Manuscript received xxxx, 2014; revised xxxx, 2015.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

rigorous alternative technique is the parametric variant of probabilistic model checking called *parametric model checking* [21], [24], [29] which symbolically or semi-symbolically computes a closed-form, perhaps highly non-linear probability function to capture the mathematical relationship between the parameters and a verification result. In practice, there may be no precise, concrete values to instantiate the parameters. It is well-known that the optimization problem of non-convex polynomial functions is NP-hard in general and even good approximate solutions are often difficult to compute using relaxation methods [36]. Another recently developed technique is the polynomial-time verification of interval-value variants of DTMCs [13], [43], [48], in which interval-value estimates of probabilities are allowed. But this technique computes optimal point-wise verification results rather than closed-form expressions with limited reusability.

In this paper, we present a novel technique of perturbation analysis for probabilistic model checking to achieve the aforementioned purpose. As in parametric model checking, our formal model is a parametric variant of a DTMC, called a *Parametric Markov Chain* (PMC), whose transition matrix contains probability variables. To cope with the imprecision with the parameter elicitation, we employ an entry-wise 1-norm to measure the *perturbation distance* of the PMC probability variables. The main technical contributions of this paper are as follows:

- We present a closed-form formulation of *asymptotic perturbation bounds* that characterize the worst effects of model perturbations on a verification result.
- We also investigate the computation problem of the two most useful forms of those bounds, namely the *linear bounds* and the *quadratic bounds*. Specifically, we investigate the mathematical programming of those two forms of bounds and the computational complexity, and then present a scalable iterative method to facilitate the numerical computation in practice.
- Lastly, we derive the *backward* counterparts of those bounds which, given a variation range for a verification result, infer the largest tolerated distance of model perturbations.

The dynamics of a DTMC is determined by a (stochastic) transition matrix. Perturbation analysis of matrix operators is a long-investigated research area which, in general terms, results in either perturbation upper bounds [49] or asymptotic expansions [33]. The former are non-asymptotic and defined in terms of a norm of the perturbed matrix, whereas the latter are approximate with increasing orders and most useful when the value of each entry in the perturbed matrix is known. Instead of directly applying existing perturbation techniques to our problem, a different perspective of our approach is the pursuit of asymptotic bounds via mathematical programming with variables measured by the 1-norm.

Informally speaking, we can identify three aspects of significance for asymptotic bounds. First, asymptotic bounds are natural theoretical metrics of the worst possible effect of the perturbed quantities on the model verification. Second, because–as mentioned—the imprecision of the parameter elicitation is usually small but not eliminated, asymptotic bounds can be used to conveniently but accurately estimate the maximum variations that might occur to a verification result. Third, the backward bounds provide an answer to the following question: How accurate should a model builder measure some specific parameters in order to safely confine a verification result within a desirable range?

For the ease of presentation, we mainly deal with the verification of extended reachability probabilities in the text, but based on automata-based verification method, our technique can also deal with ω-regular properties. We evaluate our approach with case studies on variant models of some widely studied systems, including the Google PageRank algorithm, the Zeroconf protocol and a NAND multiplexer.

The remainder of the paper is organized as follows: Section 2 presents the formal model and basic definitions. Section 3 presents the main technical results of our approach for reachability model checking. Section 4 extends those results for automata-based model checking. Section 5 discusses several issues related to the main contributions. Section 6 presents case studies. Section 7 reports related work. Section 8 concludes the paper. For readability, the appendices contain proofs for lemmas and theorems. Some preliminary results in the paper have been reported in three previous conference papers [12], [50], [52].

2 Model, 1-Norm and Example

In this section, we recall some preliminary definitions, and then present the PMC model and the 1-norm of vectors. We also present a running example based on the Google PageRank Algorithm.

2.1 Markov Chain and Preliminary Definitions

The model of Discrete-Time Markov Chains (DTMCs) or, briefly, Markov Chains (MCs) is a fundamental model that captures the probabilistic aspect of a discrete-time system.

Definition 1 (Markov Chain)**.** *An MC is a tuple* $\mathcal{M} = (S, \mathcal{P}P, \alpha, A, L)$ *where*

- S *is a finite, non-empty set of states represented as numbers* $1, \ldots, m$ *for some* $m \geq 1$,
- $\mathcal{P}$ *an* $m \times m$ transition matrix *such that, for each* $s, t \in S$, $\mathcal{P}(s,t) \in [0,1]$ *and* $\sum_{t \in S} \mathcal{P}(s,t) = 1$,
- α *an* initial distribution *such that* $\alpha(s) \geq 0$ *for each* $s \in S$ *and* $\sum_{s \in S} \alpha(s) = 1$,
- A *a set of atomic propositions, and*
- $L : S \to 2^A$ *a labeling function.*

The *digraph* of $\mathcal{M}$ is induced as follows: s is a vertex of the digraph if and only if $s \in S$, and (s,t) is an edge of the digraph if and only if $\mathcal{P}(s,t) > 0$. The size of $\mathcal{M}$, denoted as $|\mathcal{M}|$, is the sum of the numbers of vertices and edges in the digraph of $\mathcal{M}$. A *path* in $\mathcal{M}$ is an infinite sequence $\pi = s_0 s_1 s_2 \cdots$ of states in S such that $\mathcal{P}(s_i, s_{i+1}) > 0$ for each i. Denote the set of paths in $\mathcal{M}$ by $\mathrm{Path}^{\mathcal{M}}$. The probability distribution $\Pr^{\mathcal{M}}$ over $\mathrm{Path}^{\mathcal{M}}$ is defined in a standard way as in the literature (e.g., see Baier and Katoen [5, Chapter 10]). For convenience, we extend the labeling function L to paths, namely, $L(\pi) = L(s_0)L(s_1)\cdots$. We say $t \in S$ is *reachable* from s if there is a path π such that $\pi[0] = s$ and $\pi[i] = t$ for some i. Let $rch(s) \subseteq S$ denote the set of states reachable from s, and $rch(\mathcal{M}) \subseteq S$ the set of states reachable from some $s \in S$ such that $\alpha(s) > 0$.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

The problem of verifying $\mathcal{M}$ against some property φ is defined as the computation of $\Pr^{\mathcal{M}}(\varphi)$. In the sequel, we mainly deal with the following *extended reachability properties* or, briefly, *reachability properties*. Let $S_?, S_! \subseteq S$. The *property of reaching $S_!$ via $S_?$*, denoted by the conventional LTL-style "until" notation $S_?US_!$, is interpreted as the following set:

$$\{\pi \in \mathrm{Path}^{\mathcal{M}} \mid \exists i.\, \pi[i] \in S_! \wedge \forall 0 \leq j < i.\, \pi[j] \in S_?\}\ .$$

The notation $\Pr^{\mathcal{M}}(S_?US_!)$ reads as "the probability that $S_?US_!$ is satisfied by $\mathcal{M}$". If $S_? \cup S_! = S$, we abbreviate $S_?US_!$ as $\Diamond S_!$.

For simplicity, we mainly deal with reachability properties. But we demonstrate in detail later in Section 4 that our approach can be immediately generalized for the whole class of ω-regular properties.

2.2 Parametric Markov Chain and 1-Norm

A PMC model is a parametric variant of an MC with one or more undetermined transition probability variables [21]. Before presenting the PMC model, we formulate the most essential ingredients of the model. A *vector variable* $\vec{x}$ is a vector of pair-wise distinct symbolic variables $(x_1, \ldots, x_k)$ for some $k \geq 1$. Let $\mathcal{I}$ be a partition of $\{1, \ldots, k\}$. Intuitively, $\mathcal{I}$ separates $\vec{x}$ into multiple *independent* perturbed sub-vectors. To abuse notation for simplicity, we also use $\vec{x}$ to denote a vector in $\mathbb{R}^k$. Two transition matrices $\mathcal{P}$ and $\mathcal{P}'$ of the same size are structurally equivalent, denoted $\mathcal{P} \simeq \mathcal{P}'$, if they have exactly the same positions of zero entries.

We refer to the parametric counterpart of a transition matrix in a PMC as a *parametric transition matrix*. Informally, a parametric transition matrix $\mathcal{P}[\vec{x}]$ based on $\mathcal{P}$ and $\vec{x}$ is obtained by associating variables from $\vec{x}$ with some *specific* entries of $\mathcal{P}$. Formally, the (s,t)-entry of $\mathcal{P}[\vec{x}]$ is either the probability $\mathcal{P}(s,t)$ or a symbolic expression of the form $\mathcal{P}(s,t) + x_i$ for some $1 \leq i \leq k$. Here, the constant value $\mathcal{P}(s,t)$ in the symbolic expression is usually an average of a set of measured values and the variable x_i encodes the possible perturbation. We further require $\mathcal{P}[\vec{x}]$ to satisfy the following conditions, which we argue are mild and sufficient for practical purposes.

1) For all $s,t \in S$, if $\mathcal{P}(s,t) \in \{0,1\}$ then $\mathcal{P}[\vec{x}](s,t) = \mathcal{P}(s,t)$. In words, only "truly" probabilistic entries in $\mathcal{P}[\vec{x}]$ (with values larger than 0 but smaller than 1) can be parameterized.
2) For all $s,t,t' \in S$ such that $t \neq t'$, if $\mathcal{P}[\vec{x}](s,t) = a + x$ and $\mathcal{P}[\vec{x}](s,t') = b + x'$ then $x \neq x'$, namely, a single variable is not allowed to be associated with different entries in the *same* row of $\mathcal{P}[\vec{x}]$. Because entries in the same row archive outgoing transition probabilities from the same state, they cannot be parameterized with the same variable. (Note that the same x is allowed to occur in *different* rows.)
3) Let $var(s)$ be the set of variables appearing in the s^{th} row of $\mathcal{P}[\vec{x}]$. For all $s \in S$, either $var(s) = \varnothing$ or $var(s) = \{x_i\}_{i \in I}$ for some $I \in \mathcal{I}$. In words, either no variable appears in a row in $\mathcal{P}[\vec{x}]$ or variables appearing in that row form an independent sub-vector .

Moreover, whenever $\mathcal{P}[\vec{x}]$ is mentioned in the sequel, it is always assumed that $\vec{x}$ is within the following set:

$$\mathbf{U}_{\mathcal{I}} = \{\vec{x} \in \mathbb{R}^k \mid \forall I \in \mathcal{I}, \textstyle\sum_{i \in I} x_i = 0, \text{ and } \mathcal{P}[\vec{x}] \simeq \mathcal{P}\}\ .$$

The zero-sum constraint in $\mathbf{U}_{\mathcal{I}}$ expresses that the perturbation on the same-row probabilities should not distort them to be a probability distribution. The constraint of structural equivalence in $\mathbf{U}_{\mathcal{I}}$ expresses that the perturbation should not alter the structure of the original matrix.

Definition 2 (Parametric Markov Chain)**.** *A PMC is a tuple $\mathcal{M}[\vec{x}] = (S, \mathcal{P}[\vec{x}], \alpha, A, L)$ where*

- *$\mathcal{P}[\vec{x}]$ is an $m \times m$ parametric transition matrix, and*
- *all other components are the same as their counterparts in an MC (c.f., Definition 1).*

We call each variable in $\vec{x}$ a *perturbed parameter* or, simply, *parameter* of $\mathcal{M}[\vec{x}]$. We also call $\mathcal{M} = (S, \mathcal{P}, \alpha, A, L)$ the *unperturbed* MC of $\mathcal{M}[\vec{x}]$. It is easy to see that $\mathcal{M}[\vec{x}]$ with $\vec{x} \in \mathbf{U}_{\mathcal{I}}$ has the same underlaying digraph as $\mathcal{M}$. Note that Definition 2 is more restricted than the original PMC definition [21] because of the conditions added to the parametric transition matrix. But again, we believe that those conditions impose little practical restriction.

To cope with the imprecision with the parameter elicitation, we employ a vector 1-norm to measure the *perturbation distance* of $\vec{x}$ of $\mathcal{M}[\vec{x}]$. Recall that $\|\vec{x}\|_1 = \sum_{i=1}^{k} |x_i|$. Throughout the text, we write $\|\vec{x}\|_1$ as $\|\vec{x}\|$ for simplicity. The reason for choosing such a norm is two-fold. First, the 1-norm is one of the simplest norms and thus easy to use in practice. Second, compared with other norms (e.g., the Euclidean norm), the linear totality of the 1-norm results in simplified computational techniques for asymptotic perturbation bounds. We further explain the role and advantage of the 1-norm in Section 5.2.

2.3 Example: PageRank Algorithm

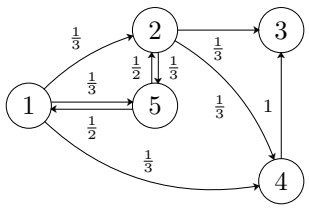
In the following, we present a running example. Consider the Google PageRank Algorithm that runs on a mini Web depicted in Figure 1a [39, Sec. 11.6].[1] Nodes in the directed graphical model refer to Web pages and edges refer to hyperlinks. The probability labeling an edge is calculated by the number of hyperlinks. For example, Web Page 1 has hyperlinks to Web Pages 2, 4 and 5, and so each of the three edges from Web Page 1 to its linked Web pages is labeled $\frac{1}{3}$. Web Page 4 is only linked to Web Page 3, and the edge from Web Page 4 to Web Page 3 is labeled 1. Web Page 3 contains no hyperlink and so has no outgoing edge. We assume the initial distribution over the five Web pages is uniform.

The directed graphical model in Figure 1a can be reformulated as a matrix $\mathcal{P}'$. The PageRank algorithm translates $\mathcal{P}'$ into a transition matrix $\mathcal{P}$ by replacing the zero rows (rows with zero entries only) of $\mathcal{P}'$ with uniform distributions. Then, the algorithm sets the PageRank probability matrix $\mathcal{P}_{\mathrm{pr}} = d\mathcal{P} + (1-d)\mathbf{1} \cdot \mathbf{v}$, where $d \in [0,1]$ is a so-called *damping factor*, row vector $\mathbf{v}$ is a so-called *personalization vector*, and $\mathbf{1}$ denotes a column vector of 1-entries. d is usually set as 0.85 but we let $d = \frac{4}{5}$ to make the presentation

1. All the concrete probabilities presented in this example are taken from the citation.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering



$$\begin{bmatrix} x_{1,1}+\frac{1}{80} & x_{1,2}+\frac{19}{60} & x_{1,3}+\frac{3}{40} & x_{1,4}+\frac{19}{60} & x_{1,5}+\frac{67}{240} \\ x_{2,1}+\frac{1}{80} & x_{2,2}+\frac{1}{20} & x_{2,3}+\frac{41}{120} & x_{2,4}+\frac{19}{60} & x_{2,5}+\frac{67}{240} \\ x_{3,1}+\frac{1}{16} & x_{3,2}+\frac{1}{4} & x_{3,3}+\frac{3}{8} & x_{3,4}+\frac{1}{4} & x_{3,5}+\frac{1}{16} \\ x_{4,1}+\frac{1}{80} & x_{4,2}+\frac{1}{20} & x_{4,3}+\frac{7}{8} & x_{4,4}+\frac{1}{20} & x_{4,5}+\frac{1}{80} \\ x_{5,1}+\frac{33}{80} & x_{5,2}+\frac{9}{20} & x_{5,3}+\frac{3}{40} & x_{5,4}+\frac{1}{20} & x_{5,5}+\frac{1}{80} \end{bmatrix}$$

(b)

Fig. 1. (a) Graphical model for a mini Web and (b) parametric transition matrix of a PMC model for PageRank

of numbers simple. $\mathbf{v}$ is set as $[\frac{1}{16}\ \frac{4}{16}\ \frac{6}{16}\ \frac{4}{16}\ \frac{1}{16}]$. With probabilistic model checking, we can calculate the probability, say, of reaching Web Pages 4 or 5 without browsing Web Page 3, namely, the probability of $\varphi_{\rm pr} = \{1,2\}U\{4,5\}$. But we also want to see the effect on such a probability result if the personalization vector is changed slightly at each Web page. To this end, we associate each entry of $\mathcal{P}_{\rm pr}$ with a variable. The resulting parametric transition matrix, denoted $\mathcal{P}_{\rm pr}[\vec{x}]$ with $\vec{x} = (x_{i,j})_{1\le i,j\le 5}$, is depicted in Figure 1b. It may be the case that $x_{i_1,j} = x_{i_2,j}$ for all i_1, i_2, j. But to achieve a general model, we let all variables be distinguished. Also note that the binary indexes of the variables are for readability. It is easy to re-index the variables to strictly follow our definition of vector variables. For example, let $x_{i,j} = x_{5i-5+j}$ for all $1 \le i, j \le 5$. The partition on $\vec{x}$ is given by $\{\{x_{i,j}\}_{j=1}^{5}\}_{i=1}^{5}$.

3 Perturbation Analysis

In this section, we present the technical details of our perturbation analysis. Section 3.1 defines a variation function. Section 3.2 analyzes asymptotic perturbation bounds, in particular, linear bounds and quadratic bounds. Section 3.3 presents the backward counterparts of asymptotic perturbation bounds.

3.1 Variation Function

Throughout this section, we focus on extended reachability properties of PMCs. Given a PMC $\mathcal{M}[\vec{x}]$ with state space S and an extended reachability property $S_?US_!$ such that $S_?, S_! \subseteq S$, for any $\vec{x} \in \mathbf{U}_{\mathcal{I}}$, we recall that $\Pr^{\mathcal{M}[\vec{x}]}(S_?US_!)$ denotes the probability that $S_?US_!$ is satisfied by $\mathcal{M}[\vec{x}]$.

The effect of perturbing $\vec{x}$ on verification of $\mathcal{M}[\vec{x}]$ against $S_?US_!$ is formally characterized by the following function, which is our main study object.

Definition 3. *A* variation function *of $\mathcal{M}[\vec{x}]$ against $S_?US_!$ is $\rho : \mathbf{U}_{\mathcal{I}} \to [0,1]$ such that*

$$\rho(\vec{x}) = \Pr^{\mathcal{M}[\vec{x}]}(S_?US_!) - \Pr^{\mathcal{M}}(S_?US_!)$$

In words, a variation function captures the difference of satisfying a reachability property by a perturbed MC and an unperturbed MC. Alternatively, a variation function can be formulated directly based on vector and matrix structures from standard probabilistic model checking (c.f., [5, Chapter 10]). To illustrate this, we define a set

$$S_0 = \{s \in (rch(\mathcal{M}) \cap S_?)\backslash S_! \mid S_! \cap rch(s) \neq \varnothing\} .$$

Let α_0 be the sub-vector of α obtained by restricting α to S_0. Let $\mathbf{A}[\vec{x}]$ be an $|S_0| \times |S_0|$ parametric matrix that contains the (possibly parameterized) transition probabilities between states in S_0, namely, $\mathbf{A}[\vec{x}](s,t) = \mathcal{P}[\vec{x}](s,t)$ for all $s,t \in S_0$. Let $\mathbf{I}$ be the identity matrix of the same size as $\mathbf{A}[\vec{x}]$. By elementary matrix theory, $\mathbf{I} - \mathbf{A}[\vec{x}]$ is invertible (and a proof is found in Baier and Katoen [5, Chapter 10]). Let $\mathbf{b}[\vec{x}]$ be a parametric vector of length $|S_0|$ that contains the probabilities of reaching $S_!$ from S_0 *in one step*, namely, $\mathbf{b}[\vec{x}](s) = \sum_{t\in S_!} \mathcal{P}[\vec{x}](s,t)$ for each $s \in S_0$. We call α_0 the *initial vector*, $\mathbf{A}[\vec{x}]$ the *constraint matrix* and $\mathbf{b}[\vec{x}]$ the *target vector* for verifying $\mathcal{M}[\vec{x}]$ against $S_?US_!$. Let $\mathbf{A}$ (resp. $\mathbf{b}$) be a matrix obtained by substituting each variable in $\mathbf{A}[\vec{x}]$ (resp. $\mathbf{b}[\vec{x}]$) to 0. The following lemma provides an alternative formulation for variation functions. Let α_0^T denote the transpose of α_0.

$$\begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix} \quad \begin{bmatrix} x_{1,1}+\frac{1}{80} & x_{1,2}+\frac{19}{60} \\ x_{2,1}+\frac{1}{80} & x_{2,2}+\frac{1}{20} \end{bmatrix} \quad \begin{bmatrix} x_{1,4}+x_{1,5}+\frac{143}{240} \\ x_{2,4}+x_{2,5}+\frac{143}{240} \end{bmatrix}$$

(a) (b) (c)

Fig. 2. (a) Initial vector, (b) constraint matrix and (c) target vector for verifying $\mathcal{M}_{\rm pr}[\vec{x}]$ against $\varphi_{\rm pr}$

Lemma 4. *For any $\vec{x} \in \mathbf{U}_{\mathcal{I}}$,*

$$\rho(\vec{x}) = \alpha_0^T(\mathbf{I} - \mathbf{A}[\vec{x}])^{-1}\mathbf{b}[\vec{x}] - \alpha_0^T(\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} .$$

We present the Taylor *expansion* of any given variation function, which is interesting by itself and is useful in the sequel. Denote $(\mathbf{I} - \mathbf{A})^{-1}$ as $\mathbf{A}^*$, $\mathbf{A}[\vec{x}] - \mathbf{A}$ as $\mathbf{A}'[\vec{x}]$, and $\mathbf{b}[\vec{x}] - \mathbf{b}$ as $\mathbf{b}'[\vec{x}]$.

Lemma 5. $\rho(\vec{x}) = \sum_{i=1}^{\infty} \rho_i(\vec{x})$ *where for each $i \ge 1$*

$$\rho_i(\vec{x}) = \alpha_0^T \overbrace{\mathbf{A}^*\mathbf{A}'[\vec{x}] \ldots \mathbf{A}^*\mathbf{A}'[\vec{x}]}^{i-1 \text{ copies of } \mathbf{A}^*\mathbf{A}'[\vec{x}]} (\mathbf{A}^*\mathbf{A}'[\vec{x}]\mathbf{A}^*\mathbf{b} + \mathbf{A}^*\mathbf{b}'[\vec{x}]) .$$

Note that some variation functions have finite expansions only. In other words, for some ρ, there is n such that $\rho_i(\vec{x}) = 0$ (for all $\vec{x} \in \mathbf{U}_{\mathcal{I}}$) if $i > n$. In this case, the largest i such that $\rho_i(\vec{x}) \neq 0$ is called *degree* of ρ.

Example. We denote the PageRank PMC model with the parametric transition matrix depicted in Figure 1b by $\mathcal{M}_{\rm pr}[\vec{x}]$ where $\vec{x} = (x_{i,j})_{1\le i,j\le 5}$. Recall that the verification problem that we are interested in for this example is the probability of $\varphi_{\rm pr}$ satisfied by $\mathcal{M}_{\rm pr}[\vec{x}]$, namely, the probability of reaching Web Pages 4 or 5 without browsing Web Page 3. We generate the corresponding initial vector, the constraint matrix and target vector as presented in Figure 2. Then, by Lemma 4, we can compute the variation function $\rho_{\rm pr}$ as presented in Figure 3, which defines the exact variation of the probability of $\varphi_{\rm pr}$ satisfied by $\mathcal{M}_{\rm pr}[\vec{x}]$ for

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

$$\rho_{\mathrm{pr}}(\vec{x}) = \begin{bmatrix}\frac{1}{5}\\ \frac{1}{5}\end{bmatrix}^T \cdot \left(\begin{bmatrix} -x_{1,1} + \frac{79}{80} & -x_{1,2} - \frac{19}{60} \\ -x_{2,1} - \frac{1}{80} & -x_{2,2} + \frac{19}{20} \end{bmatrix}^{-1} \cdot \begin{bmatrix} x_{1,4} + x_{1,5} + \frac{143}{240} \\ x_{2,4} + x_{2,5} + \frac{143}{240} \end{bmatrix} - \begin{bmatrix} \frac{79}{80} & -\frac{19}{60} \\ -\frac{1}{80} & \frac{19}{20} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \frac{143}{240} \\ \frac{143}{240} \end{bmatrix} \right)$$

Fig. 3. Closed-form variation function ρ_{pr} of $\mathcal{M}_{\mathrm{pr}}[\vec{x}]$ against φ_{pr}

any $\vec{x}$. Clearly, ρ_{pr} is a nonlinear multivariate function. Also note that some variables from $\vec{x}$ do not appear in $\rho_{\mathrm{pr}}(\vec{x})$. As mentioned before, because the values of variables in $\vec{x}$ are unknown, $\rho_{\mathrm{pr}}(\vec{x})$ sheds little light on how much of the probability of φ_{pr} satisfied by $\mathcal{M}_{\mathrm{pr}}[\vec{x}]$ will change if $\vec{x}$ is perturbed by a specific amount. Hence, we develop methods to address this issue in subsequent sections.

3.2 Asymptotic Perturbation Bound

In this subsection, we study the asymptotic bounds. Section 3.2.1 formulates the asymptotic perturbation bounds of arbitrary degrees. Section 3.2.2 and Section 3.2.3 investigate the computation and complexity of linear and quadratic asymptotic perturbation bounds, respectively. Section 3.2.5 presents the iteration methods for numerically computing the two forms of bounds.

3.2.1 Definition and Property

For $s, t \in S$, let $c_{s,t} = 1$ if $\mathcal{P}(s,t) \in \{0,1\}$ and let $c_{s,t} = \min\{\mathcal{P}(s,t), 1 - \mathcal{P}(s,t)\}$ otherwise. Let $c = \min_{s,t \in S} c_{s,t}$. The intention with the radius c is to restrict the perturbation distance of $\vec{x}$ so that the possibility of $\vec{x}$ falling out of $\mathbf{U}_{\mathcal{I}}$ is eliminated. Note that since our pursuit is the asymptotic bounds, such a restriction does not affect the analysis.

Definition 6. *Let $\rho^+, \rho^- : (0, c) \to \mathbb{R}$ such that*

$$\rho^+(\delta) = \sup\{\rho(\vec{x}) \mid \vec{x} \in \mathbf{U}_{\mathcal{I}}, \|\vec{x}\| \le \delta\}$$
$$\rho^-(\delta) = \inf\{\rho(\vec{x}) \mid \vec{x} \in \mathbf{U}_{\mathcal{I}}, \|\vec{x}\| \le \delta\} .$$

In words, given any $0 < \delta < c$, $\rho^+(\delta)$ (resp. $\rho^-(\delta)$) is the least upper bound (resp. greatest lower bound) of the variation function $\rho(\vec{x})$ subject to the condition that the distance of $\vec{x}$ is confined with δ. Intuitively, ρ^+ and ρ^- capture the largest possible effect of model perturbations on verification. However, the closed-form expressions of these exact bounds are usually difficult to obtain (see Section 5.1 for discussion). Therefore, we pursue their approximations.

Definition 7 (Asymptotic perturbation bound)**.** *A pair of upper and lower asymptotic perturbation bounds of degree n for variation function ρ are functions $f_n^+, f_n^- : (0, c) \to \mathbb{R}$ such that*

$$f_n^+(\delta) - \rho^+(\delta) = o(\delta^n)$$
$$f_n^-(\delta) - \rho^-(\delta) = o(\delta^n) ;$$

in other words,

$$\lim_{\delta \to 0} \frac{|f_n^+(\delta) - \rho^+(\delta)|}{\delta^n} = \lim_{\delta \to 0} \frac{|f_n^-(\delta) - \rho^-(\delta)|}{\delta^n} = 0 .$$

In words, Definition 7 states that, as δ tends to 0, $f_n^+(\delta)$ (resp. $f_n^-(\delta)$) converges to $\rho^+(\delta)$ (resp. $\rho^-(\delta)$) at least as fast as any polynomial function on δ of degree n. It is easy to see that ρ^+ and ρ^- themselves are upper and lower asymptotic perturbation bounds, and thus Definition 7 is legitimate. In the sequel, we often abbreviate asymptotic perturbation bounds as *asymptotic bounds*.

In general, asymptotic bounds are not unique. In the following, we present a mathematical construction of upper and lower asymptotic bounds of arbitrary degree. The construction not only provides theoretical insights but also paves the way for the computation of asymptotic bounds.

For $n \in \mathbb{N}$, we define a function $g_n^+ : (0, c) \to \mathbb{R}$ such that, for each $\delta \in (0, c)$, $g_n^+(\delta)$ is the solution of the following mathematical optimization problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_{1 \le i \le n} \rho_i(\vec{x}) \\ \text{subject to} \quad & \vec{x} \in \mathbf{U}_{\mathcal{I}} \text{ and } \|\vec{x}\| \le \delta. \end{aligned} \tag{1}$$

Similarly, $g_n^- : (0, c) \to \mathbb{R}$ is a function such that, for each $\delta \in (0, c)$, $g_n^-(\delta)$ is the solution of the following mathematical optimization problem:

$$\begin{aligned} \text{Minimize} \quad & \sum_{1 \le i \le n} \rho_i(\vec{x}) \\ \text{subject to} \quad & \vec{x} \in \mathbf{U}_{\mathcal{I}} \text{ and } \|\vec{x}\| \le \delta. \end{aligned} \tag{2}$$

Theorem 8. *For all $n \in \mathbb{N}$, g_n^+ (resp. g_n^-) is an upper (resp. lower) asymptotic bound of degree n for ρ.*

Through Theorem 8, we can see that in order to compute asymptotic bounds of order n, it suffices to consider a partial sum in the expansion of ρ up to order n. Because the constraint $\|\vec{x}\| \le \delta$ in Problems (1) and (2) can be decomposed to 2^k linear constraints, we can employ standard mathematical programming methods to compute g_n^+ and g_n^-. However, by exploiting the linear totality of $\|\cdot\|$, we present customized computational methods for asymptotic bounds of degrees one and two in the sequel.

3.2.2 Linear Perturbation Bound

In this subsection, we present a method to compute linear closed-form expressions for g_1^+ and g_1^-. Because all entries in α_0, $\mathbf{A}$, and $\mathbf{b}$ are nonnegative, and because all entries in $\mathbf{A}'[\vec{x}]$ and $\mathbf{b}'[\vec{x}]$ are either 0 or a sum expression of variables from $\vec{x}$, according to Lemma 5, we reformulate the *linear fragment* ρ_1 of ρ as follows:

$$\rho_1(\vec{x}) = \alpha_0^T \mathbf{A}^*(\mathbf{A}'[\vec{x}]\mathbf{A}^*\mathbf{b} + \mathbf{b}'[\vec{x}]) = \mathbf{h} \cdot \vec{x}$$

for some nonnegative vector $\mathbf{h} = (h_1, \ldots, h_k)$. Let $\kappa = \frac{1}{2} \max_{i,j \in I, I \in \mathcal{I}}(h_i - h_j)$.

Lemma 9. *The following equations hold:*

$$\lim_{\delta \to 0} \frac{\rho^+(\delta)}{\delta} = -\lim_{\delta \to 0} \frac{\rho^-(\delta)}{\delta} = \kappa .$$

We call κ the *condition number* of ρ. The following theorem confirms that a condition number provides an asymptotic bound of degree one.

Theorem 10. *The linear functions $\pm\kappa\delta$ are a pair of upper and lower asymptotic bounds of degree one for ρ.*

From now on, we formally refer to the linear functions $\kappa\delta$ and $-\kappa\delta$ as *linear (perturbation) bounds* for ρ.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

$$\rho_{\mathrm{pr1}}(\vec{x}) = \tfrac{11011}{66139}x_{1,1} + \tfrac{165165}{1256641}x_{1,2} + \tfrac{231}{1121}x_{1,4} + \tfrac{231}{1121}x_{1,5} + \tfrac{44759}{198417}x_{2,1} + \tfrac{223795}{1256641}x_{2,2} + \tfrac{313}{1121}x_{2,4} + \tfrac{313}{1121}x_{2,5}$$

Fig. 4. Linear fragment in the expansion of ρ_{pr}

We now consider the worst-case complexity for computing condition numbers. The generation of $\mathbf{A}$ and $\mathbf{b}$ uses a conventional graph-based algorithm. The complexity of computing the inverse of $\mathbf{I} - \mathbf{A}$ is cubic in the size of $\mathcal{M}$. Then, we have the following theorem:

Theorem 11. *Computing linear bounds (namely condition numbers) can be done in time $\mathcal{O}(|\mathcal{M}|^3)$.*

We mention in passing that it is possible to show that the computation of linear bounds is in the complexity class *probabilistic logspace*, which is believed to be lower than the complexity class P [2].

Example. For the PageRank example, a simple numerical calculation provides the expansion of the linear fragment of ρ_{pr} based on Lemma 5, as presented in Figure 4. Then, the condition number κ_{pr} is immediately calculated as $\frac{313}{2242} = 0.1396$ (with the aid of Matlab). This means that for a given small amount of model perturbations, in the worse case the probability of satisfying the property varies approximately that amount multiplied by κ_{pr}.

3.2.3 Quadratic Perturbation Bound

In this subsection, we consider the computation of quadratic closed-form expressions for g_2^+ and g_2^-. Recall that ρ_1 is the linear fragment of ρ, and accordingly, we write $\rho_1 + \rho_2$ for the *quadratic fragment* of ρ. For convenience, we introduce the concept of *directions*. A vector $\vec{v}$ is a direction of $\rho_1 + \rho_2$ if $|\vec{v}| = k$, $\sum \vec{v}_I = 0$ for all $I \in \mathcal{I}$ and $\|\vec{v}\| = 1$. So for any $\vec{x} \in \mathbf{U}_I$, $(\rho_1 + \rho_2)(\vec{x}) = \|\vec{x}\|^2\rho_2(\vec{v}) + \|\vec{x}\|\rho_1(\vec{v})$ for some direction $\vec{v}$. Informally, our strategy is to find a direction such that $\rho_1 + \rho_2$ increases or decreases at the fastest rate.

Formally, let $\vec{y}^* \in \mathbb{R}^k$ be an optimal vector of the following quadratic program:

$$\begin{array}{ll} \text{Maximize} & \rho_2(\vec{y}) \\ \text{subject to} & \sum_{i\in I} y_i = 0,\ \forall I \in \mathcal{I} \\ & \|\vec{y}\| = 1 \text{ and } \mathbf{h}\cdot\vec{y} = \kappa\ . \end{array} \tag{3}$$

Similarly, let $\vec{y}_* \in \mathbb{R}^k$ be an optimal vector of the following quadratic program:

$$\begin{array}{ll} \text{Minimize} & \rho_2(\vec{y}) \\ \text{subject to} & \sum_{i\in I} y_i = 0,\ \forall I \in \mathcal{I} \\ & \|\vec{y}\| = 1 \text{ and } \mathbf{h}\cdot\vec{y} = -\kappa\ . \end{array} \tag{4}$$

We call $\vec{y}^*$ and $\vec{y}_*$ a *maximally increasing direction* (MID) and a *maximally decreasing direction* (MDD) of $\rho_1 + \rho_2$, respectively. The following theorem confirms that MIDs and MDDs provide asymptotic bounds of degree two. Note that $\rho_1(\vec{y}^*) = -\rho_1(\vec{y}_*) = \kappa$.

Theorem 12. *The quadratic functions $\rho_2(\vec{y}^*)\delta^2 \pm \kappa\delta$ are a pair of upper and lower bounds of degree two for ρ.*

We also call $\rho_2(\vec{y}^*)\delta^2 + \kappa\delta$ (resp. $\rho_2(\vec{y}_*)\delta^2 - \kappa\delta$) an upper (resp. lower) *quadratic (perturbation) bound* for ρ.

We consider the complexity of computing quadratic bounds. First, quadratic bounds are computed based on linear bounds (or condition numbers). Second, we observe that the constraint $\|\vec{y}\| = 1$ in Problems (3) and (4) can be decomposed into linear constraints of the form $\sum_{i=1}^k \varsigma_i y_i = 1$ where $\varsigma_i \in \{-1, 1\}$. Then each of the two problems is equivalent to a combination of 2^k standard quadratic optimization problems according to different signs ς. Thus, we have the following theorem:

Theorem 13. *Computing quadratic bounds can be done in time $\mathcal{O}(poly(|\mathcal{M}|), 2^{|\vec{x}|})$.*

Alternatively, we can show that computing quadratic bounds is in the complexity class of functional NP [47].

3.2.4 Simplification by Structural Analysis

Sections 3.2.2 and 3.2.3 present the computational methods and complexity analysis for linear and quadratic bounds. In practice, the computation usually can be simplified considerably by carefully analyzing the structure of the constraint matrix and target vector. This structural analysis is based on the following three lemmas. To present (some of) these lemmas, we need some auxiliary definitions.

For each $I \in \mathcal{I}$, let $\mathbf{h}_I = (h_i)_{i\in I}$ and $\kappa_I = \max(\mathbf{h}_I) - \min(\mathbf{h}_I)$. Let $\vec{x}_I^{\min} = \{x_i \mid i \in I, h_i = \min(\mathbf{h}_I)\}$ and $\vec{x}_I^{\max} = \{x_i \mid i \in I, h_i = \max(\mathbf{h}_I)\}$. For $s \in S$, let $\mathbf{A}_s[\vec{x}]$ (resp. $\mathbf{b}_s[\vec{x}]$) denote the s^{th} row (resp. s^{th} entry) of $\mathbf{A}[\vec{x}]$ (resp. $\mathbf{b}[\vec{x}]$). Let $S_I \subseteq S$ contain exactly the states s such that $var(s) = \{x_i\}_{I\in\mathcal{I}}$. Let $\vec{v}_{i,j}$ (where $i, j \le |\vec{v}_{i,j}|$) denote a vector (i.e., direction) such that $\vec{v}_{i,j}(i) = 0.5$, $\vec{v}_{i,j}(j) = -0.5$ and $\vec{v}_{i,j}(k) = 0$ for any $k \notin \{i, j\}$.

Lemma 14. *$h_i = 0$ iff x_i does not appear in $\mathbf{A}[\vec{x}]$ or $\mathbf{b}[\vec{x}]$.*

Lemma 15. *If x_i appears in $\mathbf{b}_s[\vec{x}]$ for each $s \in S_I$, then $h_i = \max(\mathbf{h}_I)$ where $i \in I$ for all $I \in \mathcal{I}$.*

With Lemmas 14 and 15, to compute a condition number, one usually can remove the "irrelevant" variables in the constraint matrix and target vector and thus simplify the variation function.

Lemma 16. *If there are $i_\downarrow, i_\uparrow \in I$ for some $I \in \mathcal{I}$ such that*

- (c1) *$(h_{i_\uparrow} - h_{i_\downarrow})/2 = \kappa > \kappa_{I'}$ for all $I' \in \mathcal{I}\backslash\{I\}$,*
- (c2) *unless $\vec{x}_I^{\min} = \{x_{i_\downarrow}\}$, for each $s \in S_I$, all the variables in $\vec{x}_I^{\min}$ appear in $\mathbf{b}_s[\vec{x}]$ or none of them appears in $\mathbf{b}_s[\vec{x}]$ or $\mathbf{A}_s[\vec{x}]$, and*
- (c3) *unless $\vec{x}_I^{\max} = \{x_{i_\uparrow}\}$, for each $s \in S_I$, all the variables in $\vec{x}_I^{\max}$ appear in $\mathbf{b}_s[\vec{x}]$ or none of them appears in $\mathbf{b}_s[\vec{x}]$ or $\mathbf{A}_s[\vec{x}]$,*

then $\vec{v}_{i_\uparrow,i_\downarrow}$ (resp. $\vec{v}_{i_\downarrow,i_\uparrow}$) is an MID (resp. MDD) of $\rho_1 + \rho_2$.

In words, condition (c1) says that κ_I is the unique maximum in $\{\kappa_{I'}\}_{I'\in\mathcal{I}}$ and that $h_{i_\downarrow}$ (resp. $h_{i_\uparrow}$) is a minimum (resp. maximum) of $\mathbf{h}_I$. Condition (c2) (resp. (c3)) says that unless $\vec{x}_I^{\min}$ (resp. $\vec{x}_I^{\max}$) contains a single variable (equivalently, $\mathbf{h}_I$ has a unique minimum (resp. maximum) element), for *each* row of the constraint matrix and target vector either all variables from $\vec{x}_I^{\min}$ (resp. $\vec{x}_I^{\max}$) appear in the target vector only or none of them appears in the constraint matrix or target vector.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

Algorithm 1: MID and MDD computation based on Lemma 16

Input: $\mathbf{h}, \kappa, \rho_2$
if *There exist a pair* $(i_\uparrow, i_\downarrow)$ *of indices in* $I \in \mathcal{I}$ *satisfying conditions (c1) to (c3) in Lemma 16* **then**
  $\vec{y}^* \leftarrow \vec{v}_{i_\uparrow,i_\downarrow}$ and $\vec{y}_* \leftarrow \vec{v}_{i_\downarrow,i_\uparrow}$;
else
  $\vec{y}^* \leftarrow QP^*(\rho_2, \mathbf{h}, \kappa)$ and $\vec{y}_* \leftarrow QP_*(\rho_2, \mathbf{h}, \kappa)$;
  /* $QP^*()$ and $QP_*()$ are procedures for solving Problems (3) and (4) resp. */
return $\vec{y}^*$, $\vec{y}_*$

The significance of Lemma 16 is as follows. After we compute $\mathbf{h}$ and κ and know $(h_{i_\uparrow} - h_{i_\downarrow})/2 = \kappa$, if we further that the pair of indices $i_\downarrow, i_\uparrow \in I$ satisfy conditions $(c1)$ to $(c3)$ (just by scrutinizing the structure of the constraint matrix and target vector), we immediately know that $\vec{v}_{i_\uparrow,i_\downarrow}$ (resp. $\vec{v}_{i_\downarrow,i_\uparrow}$) is an MID (resp. MDD), thus avoiding the quadratic programs (i.e., Problems (3) and (4)). Based on Lemma 16, we present Algorithm 1 which can simplify the computation of MIDs and MDDs for a usual group of variation functions. Procedures $QP^*()$ and $QP_*()$ for solving Problems (3) and (4) are supported by off-the-shelf nonlinear program solvers. For example, they can be easily reformulated as constrained optimization problems in Matlab [37].

Example. By Lemmas 14 and 15, we can derive that the condition number κ_{pr} is (non-uniquely) archived by either the pair $x_{1,3}$ and $x_{1,4}$ (or $x_{1,5}$) or the pair $x_{2,3}$ and $x_{2,4}$ (or $x_{2,5}$) in the constrain matrix and target vector in Figure 2. The variation function ρ_{pr} in Figure 4 is simplified by removing all other variables, and becomes as follows:

$$\tilde{\rho}_{\text{pr1}} = \tfrac{231}{1121}x_{1,4} + \tfrac{231}{1121}x_{1,5} + \tfrac{313}{1121}x_{2,4} + \tfrac{313}{1121}x_{2,5} \ .$$

Certainly, as $\frac{231}{1121} < \frac{313}{1121}$, we conclude that κ_{pr} is achieved by $x_{2,3}$ and $x_{2,4}$ and $\kappa_{\text{pr}} = \frac{313}{1121} \cdot \frac{1}{2}$.

To save space, we do not present the expansion of the quadratic fragment ρ_{pr2} of ρ_{pr} in the text. But we note that the expansion contains 28 (symbolic) non-zero summands. Thus, re-indexing $x_{i,j}$ as x_{5i-5+j} for all $1 \le i, j \le 5$, the pair of indexes 8 and 9 meet conditions (c1) to (c3) in Lemma 16. Therefore, $\vec{v}_{9,8}$ and $\vec{v}_{8,9}$ are an MID and MDD of the quadratic fragment of ρ_{pr}, respectively. In other words, an MID (resp. MDD) is obtained by increasing (resp. decreasing) $x_{2,4}$ and decreasing (resp. increasing) $x_{2,3}$.

However, further calculations show that $\rho_{\text{pr2}}(\vec{v}_{9,8})$ and $\rho_{\text{pr2}}(\vec{v}_{8,9})$ are equal to 0. This means that the upper (resp. lower) quadratic bound of ρ_{pr} is just the linear bound $\kappa_{\text{pr}}\delta$ (resp. $-\kappa_{\text{pr}}\delta$). This is, however, not the general case. In Section 4, we will present another variation function of the PageRank model $\mathcal{M}_{\text{pr}}[\vec{x}]$ such that conditions (c1) to (c3) are satisfied but the quadratic bounds do not coincide with their linear counterparts.

3.2.5 Numerical Computation by Iteration

A realistic system model may have a large state space and a relatively high number of parameters. Like probabilistic model checking, the computation of linear and quadratic bounds can benefit from the numerical iteration, which is more efficient than the Gauss-Jordan elimination method for the inversion operation of a large matrix [25]. An iterative computation technique for linear and quadratic bounds can be envisaged from Lemma 5, and is detailed below.

Let $I_0 = \{i \in I \mid I \in \mathcal{I} \text{ and } x_i \text{ occurs in } \mathbf{A}[\vec{x}] \text{ or } \mathbf{b}[\vec{x}]\}$. Suppose $I_0 \neq \varnothing$, since otherwise the linear and quadratic bounds are trivial. For each $i \in I_0$, let $\mathbf{C}_{i,i'}$ (resp. $\mathbf{d}_{i,i'}$) be obtained from $\mathbf{A}'[\vec{x}]$ (resp. $\mathbf{b}'[\vec{x}]$) by instantiating 1 into x_i and $x_{i'}$, and by instantiating 0 into all other variables. If $i = i'$, we simply write $\mathbf{C}_i$ (resp. $\mathbf{d}_i$) instead of $\mathbf{C}_{i,i}$ (resp. $\mathbf{d}_{i,i}$). Recall that the linear coefficients in ρ are $\mathbf{h} = (h_1, \ldots, h_k)$. Then we have that

$$h_i = \tfrac{1}{2}\alpha_0^T\mathbf{A}^*(\mathbf{C}_i\mathbf{A}^*\mathbf{b} + \mathbf{d}_i), \quad i \in I_0 \tag{5}$$

and $h_i = 0$ for other $i \notin I_0$. Since $\mathbf{A}^* = \sum_{j=0}^{\infty}\mathbf{A}^j$, according to the definition of κ, it can be effectively approximated by the numerical iteration. Note that according to Lemmas 14 and 15, we may not need to compute h_i for all $1 \le i \le k$.

For quadratic bounds, we consider the lower bound only, as the upper bound can be dealt with in a symmetric manner. For each $i, i' \in I_0$ such that $i \neq i'$, let $\mathbf{E}_{i,i'}$ (resp. $\mathbf{f}_{i,i'}$) be obtained from $\mathbf{A}'[\vec{x}]$ (resp. $\mathbf{b}'[\vec{x}]$) by instantiating -1 into x_i, 1 into $x_{i'}$, and 0 into all other variables. If conditions (c1) to (c3) in Lemma 16 are satisfied, then an index pair $(i_\downarrow, i_\uparrow)$ achieving an MDD can be determined. If so, the nonlinear coefficient in the lower quadratic bound g_2^- of ρ is given by

$$\rho_2(v_{i_\downarrow,i_\uparrow}) = \tfrac{1}{4}\alpha_0^T\mathbf{A}^*\mathbf{E}_{i_\downarrow,i_\uparrow}\mathbf{A}^*(\mathbf{E}_{i_\downarrow,i_\uparrow}\mathbf{A}^*\mathbf{b} + \mathbf{f}_{i_\downarrow,i_\uparrow}) \ . \tag{6}$$

Otherwise, to invoke $QB_*()$ in Algorithm 1, we need to first compute the expression of ρ_2. Consider the coefficient of $x_{j_1}x_{j_2}$ (or, equivalently, $x_{j_2}x_{j_1}$) in ρ_2, denoted c_{j_1,j_2}, for all $1 \le j_1, j_2 \le k$. We separate three cases. For all $j_1 = j_2 \in I_0$,

$$c_{j_1,j_1} = \alpha_0^T\mathbf{A}^*\mathbf{C}_{j_1}\mathbf{A}^*(\mathbf{C}_{j_1}\mathbf{A}^*\mathbf{b} + \mathbf{d}_{j_1}) \ . \tag{7}$$

For all $j_1, j_2 \in I_0$ such that $j_1 \neq j_2$,

$$c_{j_1,j_2} = \alpha_0^T\mathbf{A}^*\mathbf{C}_{j_1,j_2}\mathbf{A}^*(\mathbf{C}_{j_1,j_2}\mathbf{A}^*\mathbf{b} + \mathbf{d}_{j_1,j_2}) - c_{j_1,j_1} - c_{j_2,j_2} \ . \tag{8}$$

If $j_1 \notin I_0$ or $j_2 \notin I_0$, then $c_{j_1,j_2} = 0$, namely, $x_{j_1}x_{j_2}$ does not occur in ρ_2. We conclude that quadratic bounds can be computed using the numerical iteration.

While probabilistic model checking uses a flat iteration to approximate $\mathbf{A}^*\mathbf{b}$, it is easy to observe that Equation (5) suggests a double-iteration for computing a condition number. In particular, we compute $\mathbf{g} \approx \mathbf{A}^*\mathbf{b}$ and then $a \approx \alpha_0^T\mathbf{A}^*\mathbf{C}_i\mathbf{g}$. Similarly, Equations (6) to (8) suggest a triple-iteration for computing the coefficients in the quadratic bound or in the quadratic fragment of the variation function. Roughly, let M denote the runtime of a flat iteration. Thus, the runtime of the numerical iteration part of probabilistic model checking is M. The runtime of iteratively computing a condition number is up to $2NM$ where $N = |I_0|$. The runtime of iteratively computing quadratic bounds is $(2N + 3)M$ if conditions (c1) to (c3) hold, and is less than $(2N + 3N^2)M$ in the worst case (because the number of non-zero quadratic coefficients is less than N^2). This analysis indicates the *scalability* of the iterative computation of condition numbers and quadratic bounds *with respect to*

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

the iterative computation as a part of probabilistic model checking. Of course, in practice, M is usually not constant but depends on factors such as the convergence rate and the termination criterion. It is also convenient to use the number of iterative steps to measure M. If so, then for a variation function with a (finite) degree M', M is bounded by M'.

3.3 Backward Analysis

In the previous sections, we have dealt with the *forward* perturbation analysis, which analyzes the worst possible consequence of model perturbations on verification results. In this subsection, we present a similar analysis in *backward* direction, which provides the maximum permitted perturbations to the model if variations of the verification result are confined to a specific range, and thus is complementary to the forward analysis. It turns out that there exists an elegant correspondence between (both exact and asymptotic) perturbation bounds and their backward counterparts.

Definition 17. *Let $\varrho^+, \varrho^- : \mathbb{R}^+ \to \mathbb{R}^+$ such that*

$$\varrho^+(\delta) = \inf\{\|\vec{x}\| \mid \vec{x} \in \mathbf{U}_{\mathcal{I}}, \rho(\vec{x}) \geq \delta\}$$
$$\varrho^-(\delta) = \inf\{\|\vec{x}\| \mid \vec{x} \in \mathbf{U}_{\mathcal{I}}, \rho(\vec{x}) \leq -\delta\} .$$

In words, given any $\delta \geq 0$, $\varrho^+(\delta)$ (resp., $\varrho^-(\delta)$) is the *smallest* perturbation distance $\|\vec{x}\|$ subject to the condition that $\vec{x} \in \mathbf{U}_{\mathcal{I}}$ and $\rho(\vec{x}) \geq \delta$ (resp. $\rho(\vec{x}) \leq -\delta$). Intuitively, whenever $\vec{x} \in \mathbf{U}_{\mathcal{I}}$ and $\|\vec{x}\| < \min\{\varrho^+(\delta), \varrho^-(\delta)\}$, $-\delta < \rho(\vec{x}) < \delta$ is guaranteed.

Note that the functions ρ^+ and ϱ^+ are both nondecreasing. The following key lemma shows that they actually form a Galois connection.

Lemma 18. *$\rho^+(\varrho^+(\delta)) \leq \delta$ and $\varrho^+(\rho^+(\delta)) \geq \delta$ for any sufficiently small $\delta \geq 0$.*

Furthermore, we prove that ϱ^+ is a pseudo-inverse of ρ^+. Similar results also hold between ϱ^- and ρ^-.

Lemma 19. *$\varrho^+\rho^+\varrho^+ = \varrho^+$ and $\rho^+\varrho^+\rho^+ = \rho^+$.*

We now present our main theorem of this subsection.

Theorem 20. 1) $\lim_{\delta\to 0} \frac{\varrho^+(\delta)}{\delta} = \lim_{\delta\to 0} \frac{\varrho^-(\delta)}{\delta} = \frac{1}{\kappa}$.

2) *Let $\hat{f}_2^+(\delta) = \delta/\kappa - \rho_2(\vec{y}^*)\delta^2/\kappa^3$ and $\hat{f}_2^-(\delta) = \delta/\kappa + \rho_2(\vec{y}_*)\delta^2/\kappa^3$. Then*

$$\lim_{\delta\to 0} \frac{|\hat{f}_2^+(\delta) - \varrho^+(\delta)|}{\delta^2} = \lim_{\delta\to 0} \frac{|\hat{f}_n^-(\delta) - \varrho^-(\delta)|}{\delta^2} = 0 .$$

Theorem 20 confirms that $1/\kappa$ serves as a *backward* condition number, while $\hat{f}_2^+$ and $\hat{f}_2^-$ are a pair of backward counterparts of quadratic bounds.

4 Automata-Based Generalization

For simplicity of presentation, in the previous sections we focused on (extended) reachability properties. In this section, we explain how our perturbation analysis of PMCs can be generalized for LTL properties and ω-regular properties via automata-based verification. We only present the necessary definitions to reveal this generalization. For completeness, we present the state-of-the-art technicalities underlying the automata-based verification of MCs in Appendix.

We first recall the syntax of LTL, which is a compact formalism for expressing (a subclass of) ω-regular properties.

Definition 21 (Linear Temporal Logic)**.** *Given a set of atomic propositions A, the syntax of LTL formulas is defined by the following rules:*

$$\varphi ::= \mathrm{tt} \mid \mathrm{a} \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U \varphi$$

where $\mathrm{a} \in A$.

Let $\Diamond\varphi$ abbreviate $\mathrm{tt}\, U\varphi$. We define a "bounded version" of $\Diamond\varphi$: Let $\Diamond^{\leq 0}\varphi$ be φ and $\Diamond^{\leq n+1}\varphi = \varphi \vee X\Diamond^{\leq n}\varphi$ for all $n \in \mathbb{N}$. The semantics of LTL is defined in a standard way by a *satisfaction relation*, denoted $\models$. Given an infinite path π of MC $\mathcal{M} = (S, \mathcal{P}, \alpha, A, L)$, and $i \in \mathbb{N}$, we define:

$$(\pi, i) \models \mathrm{tt}$$
$$(\pi, i) \models \varphi_1 \vee \varphi_2 \quad \text{iff} \quad (\pi, i) \models \varphi_1 \text{ or } (\pi, i) \models \varphi_2$$
$$(\pi, i) \models \mathrm{a} \quad \text{iff} \quad \mathrm{a} \in L(\pi[i])$$
$$(\pi, i) \models \neg\varphi \quad \text{iff} \quad (\pi, i) \not\models \varphi$$
$$(\pi, i) \models X\varphi \quad \text{iff} \quad (\pi, i+1) \models \varphi$$
$$(\pi, i) \models \varphi_1 U \varphi_2 \quad \text{iff} \quad \exists i' \geq i.\ (\pi, i') \models \varphi_2$$

Write $\pi \models \varphi$ if $(\pi, 0) \models \varphi$. The LTL-verification problem of MCs is to compute

$$\Pr^{\mathcal{M}}(\varphi) = \Pr^{\mathcal{M}}(\{\pi \in \mathrm{Path}^{\mathcal{M}} \mid \pi \models \varphi\}) .$$

The general class of ω-regular properties, including LTL properties, can be encoded by the *generalized Büchi automata* (GBA).

Definition 22 (Generalized Büchi Automata)**.** *A GBA is a tuple $\mathcal{A} = (\Sigma, Q, \Delta, Q_0, \mathcal{F})$, where*

- *Σ is a finite alphabet;*
- *Q is a finite set of states,*
- *$\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation,*
- *$Q_0 \subseteq Q$ is a set of initial states, and*
- *$\mathcal{F} \subseteq 2^Q$ is a set of acceptance sets.*

An infinite *word* $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$, if there exists an infinite *run* $\theta \in Q^\omega$ such that $\theta[0] \in Q_0$, $(\theta[i], w[i], \theta[i+1]) \in \Delta$ for $i \geq 0$ and for each $F \in \mathcal{F}$, there exist infinitely many indices $j \in \mathbb{N}$ such that $\theta[j] \in F$. Note that $w[i]$ (resp. $\theta[i]$) denotes the i^{th} letter (resp. state) of w (resp. θ). The accepted language of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of all words accepted by $\mathcal{A}$. It is well-known that GBA are expressive enough to accept the class of ω-regular languages.

For simplicity, when given an MC $\mathcal{M}$ and a GBA $\mathcal{A}$, we always assume they are *compatible*, namely, $\Sigma = 2^A$ where A is the set of atomic propositions for $\mathcal{M}$ and Σ is the alphabet of $\mathcal{A}$. Then, the automata-based verification problem is to compute

$$\Pr^{\mathcal{M}}(\mathcal{A}) = \Pr^{\mathcal{M}}(\{\pi \in \mathrm{Path}^{\mathcal{M}} \mid L(\pi) \in \mathcal{L}(\mathcal{A})\}) .$$

It is well-known that each LTL formula can be encoded by a GBA, thus $\Pr^{\mathcal{M}}(\mathcal{A})$ subsumes $\Pr^{\mathcal{M}}(\varphi)$.

The key idea of computing $\Pr^{\mathcal{M}}(\mathcal{A})$ is by constructing a *product* MC $\mathcal{M} \otimes \mathcal{A}'$ such that $\mathcal{A}'$ is a so-called *separated* GBA that is equivalent to $\mathcal{A}$ and $\Pr^{\mathcal{M}}(\mathcal{A})$ equals to $\Pr^{\mathcal{M}\otimes\mathcal{A}'}(\Diamond B)$ for some reachability problem $\Diamond B$. The formal techniques behind this idea is presented in Appendix. In the same way, given PMC $\mathcal{M}[\vec{x}]$, we can construct a product PMC

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

$$\begin{bmatrix}\frac{1}{5}\\ \frac{1}{5}\\ \frac{1}{5}\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\end{bmatrix} \quad \begin{bmatrix}0&0&0&x_{1,1}+\frac{1}{80}&x_{1,2}+\frac{19}{60}&x_{1,3}+\frac{3}{40}&0&0&0\\ 0&0&0&x_{2,1}+\frac{1}{80}&x_{2,2}+\frac{1}{20}&x_{2,3}+\frac{41}{120}&0&0&0\\ 0&0&0&x_{3,1}+\frac{1}{16}&x_{3,2}+\frac{1}{4}&x_{3,3}+\frac{3}{8}&0&0&0\\ 0&0&0&0&0&0&x_{1,1}+\frac{1}{80}&x_{1,2}+\frac{19}{60}&x_{1,3}+\frac{3}{40}\\ 0&0&0&0&0&0&x_{2,1}+\frac{1}{80}&x_{2,2}+\frac{1}{20}&x_{2,3}+\frac{41}{120}\\ 0&0&0&0&0&0&x_{3,1}+\frac{1}{16}&x_{3,2}+\frac{1}{4}&x_{3,3}+\frac{3}{8}\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\end{bmatrix} \quad \begin{bmatrix}x_{1,4}+x_{1,5}+\frac{143}{240}\\ x_{2,4}+x_{2,5}+\frac{143}{240}\\ x_{3,4}+x_{3,5}+\frac{5}{16}\\ x_{1,4}+x_{1,5}+\frac{143}{240}\\ x_{2,4}+x_{2,5}+\frac{143}{240}\\ x_{3,4}+x_{3,5}+\frac{5}{16}\\ x_{1,4}+x_{1,5}+\frac{143}{240}\\ x_{2,4}+x_{2,5}+\frac{143}{240}\\ x_{3,4}+x_{3,5}+\frac{5}{16}\end{bmatrix}$$

(a) (b) (c)

Fig. 5. (a) Initial vector, (b) constraint matrix and (c) target vector for verifying $\mathcal{M}_{\mathrm{pr}}[\vec{x}]$ against φ'_{pr}

$\mathcal{M}[\vec{x}] \otimes \mathcal{A}'$. We can verify that such a product PMC contains a parametric transition matrix satisfying the intended constraints (c.f., Section 2.2). We define a generalized variation function for $\mathcal{M}[\vec{x}]$ against $\mathcal{A}$ as

$$\rho_{\mathcal{A}}(\vec{x}) = \Pr^{\mathcal{M}[\vec{x}]}(\mathcal{A}) - \Pr^{\mathcal{M}}(\mathcal{A}) \ .$$

Then, all techniques presented in Section 3 can be lifted for $\rho_{\mathcal{A}}$ immediately.

Example. In the following, we illustrate automata-based perturbation analysis of our PageRank example. Consider the LTL formula $\varphi'_{\mathrm{pr}} = \Diamond^{\leq 3}\{4,5\}$, which informally expresses "reaching Web pages 4 or 5 within three steps (i.e., clicking the hyperlinks no more than three times)". The initial vector, constraint matrix and target vector for verifying $\mathcal{M}_{\mathrm{pr}}[\vec{x}]$ (with $\vec{x} = (x_{i,j})_{1\leq i,j\leq 5}$) against φ'_{pr} (namely $\mathcal{A}_{\varphi'_{\mathrm{pr}}}$) are presented in Figure 5. With them one immediately obtains the closed-form expression of ρ'_{pr} (similar to the one in Figure 3 for ρ_{pr}). To save space, we do not present the expression of ρ'_{pr} or its linear fragment ρ'_{pr1} or quadratic fragment $\rho'_{\mathrm{pr1}} + \rho'_{\mathrm{pr2}}$ explicitly. We note that the expansion of ρ'_{pr1} contains 15 (symbolic) summands and the expansion of ρ'_{pr2} contains 78 (symbolic) summands. We compute the condition number as 0.1443. By using Lemma 16, we can determine that an MID (resp. MDD) is obtained by increasing (resp. decreasing) $x_{3,4}$ while decreasing (resp. increasing) $x_{3,3}$ only. We further compute the quadratic bounds as $\pm 0.1443\delta - 0.0927\delta^2$. In words, for any *amount* of the perturbation δ that occurs to the model $\mathcal{M}_{\mathrm{pr}}[\vec{x}]$, using the condition number, we estimate the maximum variation of the probability of φ'_{pr} satisfied by $\mathcal{M}_{\mathrm{pr}}[\vec{x}]$ as $\pm 0.1443\delta$, and using the quadratic bounds, we estimate it as $\pm 0.1443\delta - 0.0927\delta^2$.

5 Discussion

5.1 Reflection on Linear and Quadratic Bounds

An important rationale behind our perturbation analysis is the fact that, the imprecision of quantities in the system model is usually of small-scale in the realistic situations, in other words, the model builder has various measures to narrow down the ranges of the parameter values. Consider, for example, the situation where the true value of a perturbed parameter is the expected value of a random variable. To estimate this value, we can observe the random variable to generate samples. If the sample size is large enough, by statistics theory there is a high confidence that the sample mean is sufficiently close to the true value of the parameter.

The asymptotic nature of linear and quadratic bounds implies that they are only able to provide approximations rather than exact bounds. Nonetheless, for stochastic systems with parameters subject to small but nontrivial perturbations, linear and quadratic bounds provide adequate estimates and fulfil our requirements to a satisfactory degree in application (as shown later in Section 6). Moreover, linear and quadratic bounds have two advantages. First, they enjoy simple closed forms that uniformly characterize the sensitivity and robustness of a verification result, regardless of the actual model perturbation. Second, their computation, in theory, has relatively low worst-case complexity compared with the point-wise computation of exact perturbation bounds and, in practice, can use the numerical iteration algorithm as their point-wise counterparts [12].

It is natural to expect that asymptotic bounds of higher degrees provide more accurate approximations, but at cost of high computational burden. This challenging generalization is left to future work.

5.2 Reflection on Vector Norm

Many results presented in Section 3 depend on the 1-norm of the perturbed parameters. In short, by choosing such a norm, the condition number can be computed by linear programming, and the quadratic bounds are computed by pursuing an MID and an MDD, which are computed largely based on the condition number. In what follows, we clarify the (in)dependence in more detail.

Certainly, the variation function ρ is independent of any norm. We denote the resulted upper bound as $\rho^{+}_{\|\cdot\|_*}$ if the norm $\|\cdot\|_*$ is adopted. Because the lower bound is symmetric to the upper bound, we only need to discuss the upper bound here. As we only consider finite norms, a well-known fact from the matrix theory states that there are positive constants c and C such that

$$c\|\vec{x}\| \leq \|\vec{x}\|_* \leq C\|\vec{x}\|, \quad \vec{x} \in \mathbb{R}^k. \tag{9}$$

Hence, one can easily show that Theorem 8 holds for $\rho^{+}_{\|\cdot\|_*}$ as well if Problem (1) and (2) are adapted for $\|\cdot\|_*$.

However, the computation of linear bounds based on an arbitrary norm *cannot* directly resort to linear programming. To see this, consider the following simple variation function:

$$\rho_{\mathrm{eg}}(x,y,z) = x/2 + y/3 + x^2/4 \ .$$

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

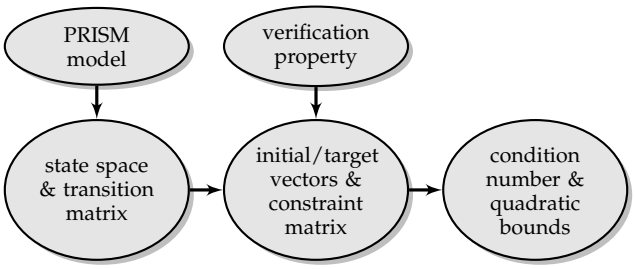


Fig. 6. Computation procedure in the experiment

Note that ρ_{eg} is obtained from a simple PMC with the reachability problem. (The detailed model is omitted for simplicity.) Assume the Euclidean norm $\|\cdot\|_2$ to measure the variables of ρ_{eg}. By Theorem 8, a linear upper bound of ρ_{eg} is an optimal solution of the following problem:

$$\begin{array}{ll}\text{Maximize} & x/2 + y/3 \\ \text{subject to} & x + y + z = 0 \text{ and } \sqrt{x^2+y^2+z^2} \leq \delta.\end{array}$$

The above problem is clearly *not* a linear program. Its *unique* optimal solution is as follows:

$$x = \tfrac{4}{\sqrt{42}}\delta, \quad y = \tfrac{1}{\sqrt{42}}\delta, \quad z = -\tfrac{5}{\sqrt{42}}\delta\ .$$

Thus, the linear upper bound is $7\delta/(3\cdot\sqrt{42})$. Furthermore, the linear bounds say little about the quadratic bounds. To see this, replace the objective function with the quadratic function ρ_{eg} itself. It can be verified that the above solution is *not* an optimal solution of the modified problem. This example demonstrates the dependence of the computational techniques in Sections 3.2.2 and 3.2.3 on the 1-norm.

It is worthwhile to point out that, Inequality (9) implies an inequality between an exact bound based on the 1-norm and one based on another norm. In particular, as $\sqrt{k}\|\vec{x}\| \leq \|\vec{x}\|_2 \leq \|\vec{x}\|$ and ρ^+ is an increasing function, we have that

$$\rho^+(\delta) \leq \rho^+_{\|\cdot\|_2}(\delta) \leq \rho^+(\sqrt{k}\delta)\ .$$

The above bounding relationship can be used to provide (loose) estimates for our asymptotic bounds if the Euclidean norm is adopted. Note that for other norms (e.g., the maximum norm $\|\cdot\|_\infty$), a similar relationship holds.

6 Case Studies

In this section, we evaluate the applicability of the two forms of asymptotic bounds, namely, condition numbers and quadratic bounds. We mainly consider the *accuracy* of these bounds. Our objective is to demonstrate that, despite the asymptoticity, these bounds can be used to accurately predict the actual worst effect caused by small but non-trivial model perturbations. Our computation of these bounds in the case studies adopts the numerical iteration. We have analyzed the scalability of this iterative computational method with respect to the one in probabilistic model checking in Section 3.2.5. In this section, we also provide empirical evidence for scalability through analysis of computation runtime.

The case studies are based on the PageRank example and two other benchmarks, namely a Zeroconf protocol model and a NAND multiplexer model. We have implemented our iterative computational method in Matlab and interacted with PRISM. The procedure is depicted in Figure 6. We first specify a system model in PRISM and export its state space and transition matrix into matrices in Matlab. We then generate the constraint matrix and target vector with respect to a verification property, namely a reachability property or LTL property. Finally, we calculate the condition number and quadratic bounds using my implementation prototype. All PRISM specifications and the Matlab source codes are available at the first author's Web site.[2]

TABLE 1
Accuracy test data for PageRank w.r.t. (a) φ_{pr} and (b) φ'_{pr}

pert.	result (10^{-4})		by CN (10^{-4})
	upper	lower	
0.000	6891.4659207	6891.4659207	+/-0.000000000
0.005	+6.980374536	-6.980374536	+/-6.980374665
0.010	+13.96074907	-13.96074907	+/-13.96074933

(a)

pert.	result (10^{-4})		by CN (10^{-4})	by QB (10^{-4})	
	upper	lower		upper	lower
0.000	9038.7	9038.7	+/-0.000	+0.000	-0.000
0.005	+7.189	-7.236	+/-7.213	+7.189	-7.236
0.010	+14.33	-14.52	+/-14.43	+14.33	-14.52

(b)

6.1 PageRank Algorithm

Recall that the two verification properties for the PageRank model $\mathcal{M}_{\mathrm{pr}}[\vec{x}]$ that we considered earlier are $\varphi_{\mathrm{pr}} = \{1,2\}U\{4,5\}$ and $\varphi'_{\mathrm{pr}} = \Diamond^{\leq 3}\{4,5\}$, and the condition numbers (CN for short) and quadratic bounds (QB for short) corresponding to these two properties are as follows:

property	CN	QB
φ_{pr}	$\kappa_{\mathrm{pr}} = 0.1396$	$\pm 0.1396\delta$
φ'_{pr}	$\kappa'_{\mathrm{pr}} = 0.1443$	$\pm 0.1443\delta - 0.0927\delta^2$

Our computation reveals that the maximum variations of the probability of satisfying φ_{pr} almost overlays with $\kappa_{\mathrm{pr}}\delta$ for any perturbation distance δ. This is not the case for φ'_{pr}. But our validation data as presented in Table 1 shows that up to the perturbation distance of 0.01, the condition number and quadratic bounds can accurately estimate the maximum variations. When the perturbation distance is 0, the upper and lower bounds of the probabilistic results coincide with the unperturbed results. Note that the 0.01 perturbation distance is a nontrivial distance compared with

2. http://www.comp.nus.edu.sg/~sugx/tse15/

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

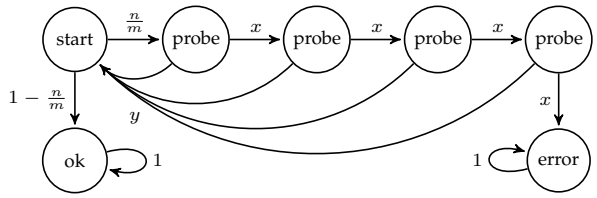


Fig. 7. Zeroconf protocol with uncertain message loss rate

TABLE 2
Accuracy test data for Zoreconf protocol

loss rate	result (10^{-6})	%	estimated (10^{-6})	
			CN	QB
0.095	-36.73	-19.8%	-39.97	-35.73
0.096	-29.89	-16.9%	-31.97	-29.26
0.097	-22.79	-12.3%	-23.98	-22.46
0.098	-15.47	-8.33%	-15.99	-15.31
0.099	-7.859	-4.23%	-7.993	-7.824
0.100	185.67	–	–	–
0.101	+8.131	+4.38%	+7.993	+8.162
0.102	+16.54	+8.91%	+15.99	+16.66
0.103	+25.22	+13.6%	+23.98	+25.50
0.104	+34.20	+18.4%	+31.97	+34.68
0.105	+43.48	+23.4%	+39.97	+44.20
0.095	-6.845	-4.39%	-6.966	-6.813
0.096	155.79	–	–	–
0.097	+7.089	+4.54%	+6.966	+7.120
0.103	-8.979	-4.08%	-9.127	-8.941
0.104	219.87	–	–	–
0.105	+9.277	+4.22%	+9.127	+9.313

the smallest constant transition probability $\frac{1}{80} = 0.0125$ in the parametric transition matrix of $\mathcal{M}_{\text{pr}}[\vec{x}]$ (see Figure 1b). We also see that the quadratic bounds provide more accurate estimates than the condition number.

6.2 Zeroconf Protocol

Consider the IPv4 Zeroconf protocol implemented in some physical network with noisy communication channels. The Zeroconf protocol enables a new host to join a computer network automatically and with "zero configuration" (such as without pre-assignment of an IP address). Figure 7 depicts a lightweight abstract model of Zeroconf that uses a maximum of four message probes for the new host to discover an unused IP address. At the *start* state, the new host randomly selects an IP address and either moves to a *probe* state or the *ok* state, depending on whether the selected IP address is occupied or not. At each of the four *test* states, the new host sends a probe to existing hosts and waits. If it receives a reply within a specified time, then the process goes back to the *probe* state; if not, the process proceeds to the next *probe* state or the *error* state. The constant number n is the number of existing hosts, and $m = 60{,}534$ is the size of the IP address space as specified in Zeroconf. The variable x refers to the loss rate of a probe or its reply. Note that $y = 1 - x$. In reality, x relies on an ad-hoc statistical estimation and is instantiated by the sample mean $\bar{x}$. Due to sampling errors or environmental influences, there may be some perturbations on x.

We are interested to know the probability that an address collision happens. This problem can be stated as the reachability property $\Diamond\{error\}$. For experimentation purposes, we assume that the number of hosts in the network is 35,000, and let the estimated message loss rate be 0.1 while the actual rate be perturbed up to $\pm 5\%$. We calculate the condition number and quadratic bounds as follows:

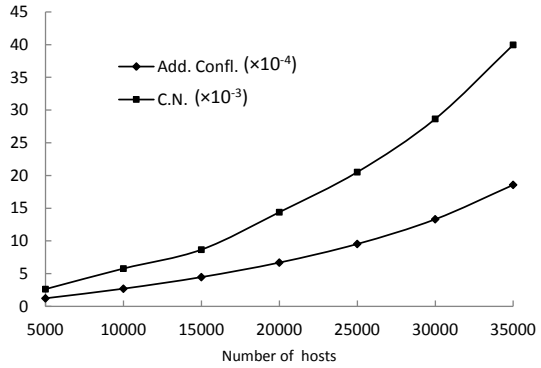


Fig. 8. Address collision and its sensitivity w.r.t. host No.

loss rate	CN ($\times 10^{-3}$)	QB ($\times 10^{-3}$)
0.100	3.9965	$\delta^2 \cdot 42.308 \pm \delta \cdot 3.9965$

The experimental data for the accuracy test are presented in Table 2, from which we observe that

- there are non-negligible effects on the address collision probability due to small perturbations of the message loss rate;
- the pair of quadratic bounds provide more accurate estimates than the condition number; and
- as the message loss rate increases or decreases from the estimated rate, the deviation between the actual result and the estimated result increases.

To further test the last observation presented above, we perform the same perturbation analysis pivoted at 0.096 and 0.104 message loss rates. The condition numbers and quadratic bounds are as follows:

loss rate	CN ($\times 10^{-3}$)	QB ($\times 10^{-3}$)
0.096	3.4832	$\delta^2 \cdot 38.336 \pm \delta \cdot 3.4832$
0.104	4.5634	$\delta^2 \cdot 46.539 \pm \delta \cdot 4.5634$

The additional experimental data are also presented in Table 2. By comparing the three groups of data in the table, we observe that the perturbation bounds provide more accurate estimates when the perturbation distance of the message loss rate is small. We also test the dependence relationship of the address collision probability and the condition number on the number of existing hosts. Two increasing trends are depicted in Figure 8.

6.3 NAND Multiplexing

Multiplexing is a technique for building more reliable components from less reliable ones. Figure 9 depicts an im-

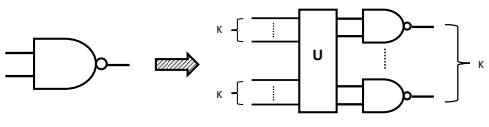


Fig. 9. NAND multiplexer

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

TABLE 3
Accuracy test data for NAND multiplexer with (a) perturbed input and (b) perturbed error rate

input	result	%	estimated by	
			CN	QB
0.884	-.12710	-14.9%	-.10771	-.12842
0.888	-.09208	-10.8%	-.08078	-.09243
0.892	-.05903	-6.91%	-.05385	-.05903
0.896	-.02825	-3.31%	-.02693	-.02822
0.900	0.85413	–	–	–
0.904	+.02555	+2.99%	+.02693	+.02563
0.908	+.04828	+5.65%	+.05385	+.04867
0.912	+.06816	+7.98%	+.08078	+.06913
0.916	+.08522	+9.97%	+.10771	+.08699

(a)

error rate	result	%	estimated by	
			CN	QB
0.006	+.00929	+1.09%	+.00970	+.00952
0.007	+.00670	+0.82%	+.00728	+.00717
0.008	+.00469	+0.55%	+.00485	+.00480
0.009	+.00236	+0.28%	+.00243	+.00241
0.010	0.85413	–	–	–
0.011	-.00238	-0.28%	-.00243	-.00244
0.012	-.00478	-0.56%	-.00485	-.00490
0.013	-.00720	-0.84%	-.00728	-.00738
0.014	-.00964	-1.13%	-.00970	-.00988

(b)

perfect NAND gate and a NAND multiplexer. The NAND multiplexer is devised by replicating the NAND gate K times. Since the multiplexer is a component of a system that may contain other unreliable components, the inputs for the multiplexer are two bundles of K logical values 1 (representing a stimulated result) or 0 (representing a non-stimulated result) as determined by a probability distribution. The functionality of the U unit is to randomly choose two input values as the inputs for the NAND gates. It is assumed that the NAND gates have the same error rate and that they fail independently. Whether the overall output of the multiplexer is stimulated or not depends on the number of stimulated outputs of the NAND gates. More specifically, we specify a small value $0 < \Delta < 0.5$. Then, the overall output of the multiplexer is considered to be stimulated if at least $K(1-\Delta)$ of the outputs of the NAND gates are stimulated, and non-stimulated if no more than $K\Delta$ of them are stimulated. In the case that neither of the two conditions is satisfied, the overall output is undecided.

The NAND multiplexer specified in PRISM has two probability parameters, namely the original stimulated input and the error rate. We analyze the consequence of perturbations to the two parameters on the overall stimulated or non-stimulated probabilities of the multiplexer. In our experiments, we set $K = 40$ (resulting in 6,642 states in the model) and $\Delta = 0.25$. We also set the unperturbed probability of stimulated inputs as 0.9 and the unperturbed gate error rate as 0.01. Tables 3 and 3 present validation data corresponding to the two parameters. Again, similar to the previous two case studies, we observe both non-trivial variations on verification results and accurate estimates when the two parameters are perturbed slightly.

TABLE 4
Runtime analysis data in all case studies

model	#st.	#pa.	verification		CN		QB	
			ms	#it.	ms	#it.	ms	#it.
PR	25	25	0.39	13	0.70	26	–	–
PR′	25	25	0.19	3	0.28	7	0.33	10
ZCF	7	2	2.40	87	5.40	184	8.20	289
NAND	6642	2	8.55	161	20.0	321	30.0	480

6.4 Runtime Analysis

We summarize runtime results of our experiment in all case studies together with the model information for each example in Table 4. Note that we only consider the iteration part of computation. The termination criterion for all iterations is set as 10^{-12}. The machine that we used to run the experiment is an MS Windows 7 desktop with 3.4GHz quad-core CPU and 16GB RAM in total. We adopt two runtime measures, namely the actual elapsed time and the iterative number. It is emphasized that we do not aim to devise an optimized implementation but only to evaluate the scalability of the iteration for computing CN and QB with respect to that for computing a probabilistic verification outcome. From our analysis results, a reasonable scalability is demonstrated via a comparison of data in the the correspondent columns of the table.

6.5 Summary

In summary, from the previous case studies, we learn that

- small but nontrivial perturbations on probability parameters of the stochastic system model can cause non-negligible variations on verification, potentially turning acceptable results into unacceptable ones;
- condition numbers and quadratic bounds can provide sufficiently accurate estimates of the maximum variations of the verification results;
- as the perturbation distance increases, estimates by quadratic bounds are tighter than those by condition numbers; and
- the runtime analysis manifests promising scalability for the iterative computation of those bounds.

7 Related Work

7.1 Parametric Model Checking

One key definition underlying our approach is the probability function $\Pr^{\mathcal{M}[\vec{x}]}(S_?US_!)$ (c.f., Definition 3) for a given PMC $\mathcal{M}[\vec{x}]$ and a reachability property $S_?US_!$. Different methods for generating such a probability function are studied in the literature. Daws [21] presented a language-theoretic method to compute the exact rational expression of such a probability function, based on the fact that all paths satisfying φ can be represented as a (finite) Büchi automaton. Once such an automaton is constructed, one can use a standard method in automata theory to infer a regular expression, which is further evaluated to the rational expression of the function. Hahn et al. [29] improved the efficiency of Daws's method for most practical cases by reducing the state space and by using a method called early evaluation,

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

even though the length of the rational expression in the worst case is unchanged, namely $\Theta(n^{\log n})$ where n is the size of the state space of the PMC. Their parametric model checking also deals with rewards properties and MDPs, and is implemented in a tool called PARAM [29].

Filieri et al. [24] presented a parameterized version of the matrix inversion operation, namely the Gauss-Jordan elimination method, to compute the same probability function. Since the transition matrix of the PMC model is usually sparse in practice, the method by Filieri et al. [24] leads to a reasonable computational cost. The worst-case complexity is $\mathcal{O}(n^3 \cdot \tau^c)$ where n is the size of the state space of the PMC, τ is the average number of outgoing transitions from each state (thus $\tau \ll n$ by sparsity), and c is the number of rows containing symbolic entries. Their method can also deal with properties expressed by nested Probabilistic Computation Tree Logic (PCTL) formulas, which cannot be directly represented as finite automata. They also presented sensitivity analysis directly using the first-order partial derivatives of the probability function.

Our perturbation analysis is in contrast to these existing approaches mainly on two aspects. First, we consider the Taylor expansion of the variation function but not the closed-form probability function. In practice, we are particularly concerned with the linear and quadratic fragment of the expansion, we can employ the iterative numerical computation and avoid the expensive symbolic or semi-symbolic computation in parametric model checking. Second, to provide an outcome of verification or sensitivity analysis, these approaches require concrete numerical values to be instantiated into the variables of the probability function, instead of using a norm to measure those variables. Although—not explicitly mentioned in those papers—one can exploit an optimization method to deal with the worst effect of the imprecise variables on the probability function, it is well-known that the optimization problem of non-convex polynomial functions is NP-hard and even good approximate solutions are difficult to compute using the relaxation methods (e.g, semidefinite programing [36]).

Like Daws [21] and Hahn et al. [29] but unlike Filieri et al. [24], we do not address nested PCTL formulas. The reason is two-fold. First, albeit nested PCTL formulas are of interest in theory, ω-regular properties are expressive enough to represent most interesting temporal properties for real-world system models. Second, nested PCTL formulas break the continuity of the probability function (as demonstrated in our previous work [51]) and thus limit the applicability of condition numbers and quadratic bounds.

There are research works [6], [13], [22] on parameter estimations and model repairs, which in general address how to determine the values of some parameters in the Markov model or to fix the model such that a given, originally unsatisfied temporal property becomes satisfied. Those works are complementary to parametric model checking and our perturbation analysis.

We also mention that parameters in a PMC are described with probability distributions in some papers [9], [38]. The authors employed statistical inference [9] or simulation [38] to deal with the verification problem of the resulted model. By contrast, the reasoning techniques adopted by us and the aforementioned literature are analytical.

7.2 Model Checking with Uncertain Probabilities

In the probabilistic model checking setting, there is a line of research on Markov models with uncertainties. The uncertainties with the probabilities in the transition matrix of the model are characterized by interval values. Sen et al. [48] presented two semantic interpretations for an interval-valued variant of DTMCs in which the uncertain probabilities in a transition matrix are given as intervals. Such a model is interpreted either as a set of DTMCs, called an Uncertain Markov Chain (UMC), or as a variant of a Markov Decision Process (MDP) with an (uncountably) infinite set of adversaries, called an Interval-valued MDP (IMDP). Sen et al. [48] considered the complexity bounds of the model checking problems for the two kinds of models (namely UMCs and IMDPs) against PCTL. Chatterjee et al. [11] considered the problems against an extended logic of PCTL, denoted ω-PCTL, which can express all ω-regular properties, and presented tighter complexity bounds that those by Sen et al. [48]. Benedikt et al. [7] considered the problem for IMDP against LTL, which, despite a fragment of ω-PCTL, leads to a different complexity upper bound. Chen et al. [15] presented complexity bounds (namely P-completeness) that further improve the results by Chatterjee et al. [11] on the model checking problem for IMDPs against PCTL, and a different complexity lower bound for the problem for UMC against PCTL. Puggelli et al. [43] presented P-completeness complexity bounds for an generalization of IMDPs that use convex sets to characterize the uncertain probabilities.

The methods presented in the above works are related to the perturbation bounds of variations on verification results that we address in this paper. Indeed, this relationship has been exploited in our previous work [12] to compute *point-wise exact* bounds for the probability function. However, point-wise bounds are not in closed form, and thus are less informative and useful for characterizing the consequences of model perturbations on its verification if the value ranges of the parameters are unknown.

Another similar work in the same setting is the approximate model checking method based on interval and affine arithmetic for UMCs proposed by Ghorbal et al. [27]. Such a method computes over-approximate verification bounds that also are not in closed form.

7.3 Perturbation Analysis of Matrix Operator

Since the dynamics of a DTMC (and a PMC) is determined by a stochastic transition matrix, the problem of the variation function that we investigate in this paper can be alternatively and equivalently defined using the inversion of (a fragment of) the transition matrix (c.f., Lemma 4). In view of this, our paper is in line with a long-investigated research area called perturbation analysis of operators on matrices (e.g., inversion, rank, eigenvalue and stationary vector). The existing literature in this area usually provide two results, namely perturbation upper bounds [17], [49] and asymptotic expansions [33]. The upper bounds are non-asymptotic and defined in terms of a norm of the perturbed matrix, whereas the expansions are approximate but most useful when the value of each entry in the perturbed matrix is known. Because of requiring quite different mathematical techniques, these two are usually studied separately

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

in the literature. But both are useful in practice, such as in numerical computation and dynamic robust control. In software engineering, in particular, the partial derivatives (which are equivalent to the linear fragment of the Taylor expansion) have been used to analyze the sensitivity of the overall system performance (e.g., reliability) to a parameter belonging to some system component [16], [18], [23].

We do not directly exploit an exiting technique to deal with our problem in the context of probabilistic model checking. Our pursuit of asymptotic bounds is clearly related but in contrast to the upper-bound approach and the asymptotic approach. On the one hand, we make use of the Taylor expansion of our variation function. On the other hand, we measure the perturbed parameters by a vector norm and pursue asymptotic bounds as mathematical programming problems. Moreover, our approach emphasizes the computation of those bounds and considers both the computational complexity and the iteration-based numerical computation.

8 Conclusions

In this paper, we have presented a systematic approach to formulate and compute asymptotic perturbation bounds, especially the linear and quadratic forms of those bounds, to support probabilistic model checking applied to systems containing empirically determined probability parameters. We showed the advantage and significance of those bounds through both theoretical analysis and empirical evaluation.

Future research directions fall into two categories. The first one includes the tool support and applications of our approach. For experimentation purposes, we have developed an implementation prototype. A more sophisticated, self-contained toolkit is an important part of our further work. We also plan to apply our approach to analyzing specific problems in software engineering, such as decision-making of self-adaptive systems based on imprecise parameter estimations. The second work category includes various topics of theoretical and methodological enhancements for our approach, such as perturbation analysis for CTMCs.

Acknowledgment

This research is partially supported by the Singapore Ministry of Education under grant R-252-000-458-133, the Australian Research Council under grant DP130102764, the National Natural Science Foundation of China (Grant Nos. 61428208 and 61472412), the CAS/SAFEA International Partnership Program for Creative Research Team, and an oversea grant from the State Key Laboratory of Novel Software Technology at Nanjing University. We thank the anonymous reviewers for instructive comments on an earlier version of the paper.

References

[1] M. Agrawal, S. Akshay, B. Genest, and P. S. Thiagarajan, "Approximate verification of the symbolic dynamics of markov chains," *Journal of ACM*, vol. 62, no. 1, pp. 2:1–2:34, Mar. 2015.

[2] E. Allender and M. Ogihara, "Relationships among PL, #L, and the determinant," *ITA*, vol. 30, no. 1, pp. 1–21, 1996.

[3] R. Alur, S. La Torre, and P. Madhusudan, "Perturbed timed automata," in *Proceedings of the 8th international conference on Hybrid Systems: computation and control*, ser. HSCC'05. Springer-Verlag, 2005, pp. 70–85.

[4] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-checking algorithms for continuous-time markov chains," *Software Engineering, IEEE Transactions on*, vol. 29, no. 6, pp. 524–541, 2003.

[5] C. Baier and J.-P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.

[6] E. Bartocci, R. Grosu, P. Katsaros, C. R. Ramakrishnan, and S. A. Smolka, "Model repair for probabilistic systems," in *Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 326–340.

[7] M. Benedikt, R. Lenhardt, and J. Worrell, "LTL model checking of Interval Markov Chains," in *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 32–46.

[8] P. Bouyer, N. Markey, and P.-A. Reynier, "Robust model-checking of linear-time properties in timed automata," in *Proceedings of the 7th Latin American conference on Theoretical Informatics*, ser. LATIN'06. Springer-Verlag, 2006, pp. 238–249.

[9] R. Calinescu, C. Ghezzi, K. Johnson, M. Pezzé, Y. Rafiq, and G. Tamburrelli, "Formal verification with confidence intervals: A new approach to establishing the Quality-of-Service properties of software systems," *Reliability, IEEE Transactions on*, 2015.

[10] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola, "Self-adaptive software needs quantitative verification at runtime," *Commun. ACM*, vol. 55, no. 9, pp. 69–77, Sep. 2012.

[11] K. Chatterjee, K. Sen, and T. A. Henzinger, "Model-checking omega-regular properties of Interval Markov Chains," in *FoSSaCS*, ser. Lecture Notes in Computer Science, R. M. Amadio, Ed., vol. 4962. Springer, 2008, pp. 302–317.

[12] T. Chen, Y. Feng, D. S. Rosenblum, and G. Su, "Perturbation analysis in verification of discrete-time markov chains," in *Proceedings of the 25th International Conference on Concurrency Theory (CONCUR'14)*, 2014.

[13] T. Chen, E. Hahn, T. Han, M. Kwiatkowska, H. Qu, and L. Zhang, "Model repair for markov decision processes," in *Theoretical Aspects of Software Engineering (TASE), 2013 International Symposium on*, July 2013, pp. 85–92.

[14] T. Chen, T. Han, J.-P. Katoen, and A. Mereacre, "LTL model checking of time-inhomogeneous Markov chains," in *Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis*, ser. ATVA'09. Springer-Verlag, 2009, pp. 104–119.

[15] T. Chen, T. Han, and M. Z. Kwiatkowska, "On the complexity of model checking interval-valued discrete time Markov chains," *Inf. Process. Lett.*, vol. 113, no. 7, pp. 210–216, 2013.

[16] R. C. Cheung, "A user-oriented software reliability model," *IEEE Trans. Softw. Eng.*, vol. 6, no. 2, pp. 118–125, Mar. 1980.

[17] G. E. Cho and C. D. Meyer, "Comparison of perturbation bounds for the stationary distribution of a Markov chain," *Linear Algebra Appl*, vol. 335, pp. 137–150, 2000.

[18] V. Cortellessa and V. Grassi, "A modeling approach to analyze the impact of error propagation on reliability of component-based systems," in *Proceedings of the 10th International Conference on Component-based Software Engineering*, ser. CBSE'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 140–156.

[19] C. Courcoubetis and M. Yannakakis, "The complexity of probabilistic verification," *J. ACM*, vol. 42, no. 4, pp. 857–907, 1995.

[20] J.-M. Couvreur, N. Saheb, and G. Sutre, "An optimal automata approach to LTL model checking of probabilistic systems," in *LPAR*, 2003, pp. 361–375.

[21] C. Daws, "Symbolic and parametric model checking of discrete-time Markov chains," in *Proceedings of the First International Conference on Theoretical Aspects of Computing*, ser. ICTAC'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 280–294.

[22] R. Donaldson and D. Gilbert, "A model checking approach to the parameter estimation of biochemical pathways," in *Proceedings of the 6th International Conference on Computational Methods in Systems Biology*, ser. CMSB'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 269–287.

[23] A. Filieri, G. Tamburrelli, and C. Ghezzi, "Supporting self-adaptation via quantitative verification and sensitivity analysis at run time," *Software Engineering, IEEE Transactions on*, In Press.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

[24] A. Filieri, C. Ghezzi, and G. Tamburrelli, "Run-time efficient probabilistic model checking," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE'11. New York, NY, USA: ACM, 2011, pp. 341–350.

[25] V. Forejt, M. Z. Kwiatkowska, G. Norman, and D. Parker, "Automated verification techniques for probabilistic systems," in *SFM*, ser. Lecture Notes in Computer Science, M. Bernardo and V. Issarny, Eds., vol. 6659. Springer, 2011, pp. 53–113.

[26] C. Ghezzi, L. S. Pinto, P. Spoletini, and G. Tamburrelli, "Managing non-functional uncertainty via model-driven adaptivity," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE'13. IEEE Press, 2013, pp. 33–42.

[27] K. Ghorbal, P. S. Duggirala, V. Kahlon, F. Ivancic, and A. Gupta, "Efficient probabilistic model checking of systems with ranged probabilities," in *Reachability Problems - 6th International Workshop*, 2012, pp. 107–120.

[28] E. M. Hahn, T. Han, and L. Zhang, "Synthesis for PCTL in parametric Markov Decision Processes," in *NASA Formal Methods*, ser. Lecture Notes in Computer Science, M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, Eds., vol. 6617. Springer, 2011, pp. 146–161.

[29] E. Hahn, H. Hermanns, and L. Zhang, "Probabilistic reachability for parametric Markov models," *International Journal on Software Tools for Technology Transfer*, vol. 13, no. 1, pp. 3–19, 2011.

[30] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal aspects of computing*, vol. 6, no. 5, pp. 512–535, 1994.

[31] B. Heidergott, "Perturbation analysis of Markov chains," in *WODES 2008. 9th International Workshop on Discrete Event Systems*, 2008, pp. 99–104.

[32] D. Kähler and T. Wilke, "Complementation, disambiguation, and determinization of Büchi automata unified," in *ICALP (1)*, ser. Lecture Notes in Computer Science, L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, Eds., vol. 5125. Springer, 2008, pp. 724–735.

[33] T. Kato, *Perturbation Theory for Linear Operators (2ed)*. Berlin, Heidelberg, New York: Springer-Verlag, 2005.

[34] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Springer, 2011, pp. 585–591.

[35] ——, "The PRISM benchmark suite," in *Proc. 9th International Conference on Quantitative Evaluation of SysTems (QEST'12)*. IEEE CS Press, 2012, pp. 203–204.

[36] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM J. on Optimization*, vol. 11, no. 3, pp. 796–817, Mar. 2000.

[37] MATLAB, *version 8.0 (R2012b)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[38] I. Meedeniya, I. Moser, A. Aleti, and L. Grunske, "Evaluating probabilistic models with uncertain model parameters," *Software and Systems Modeling*, vol. 13, no. 4, pp. 1395–1415, 2014.

[39] P. V. Mieghem, *Performance Analysis of Communications Networks and Systems*. New York, NY, USA: Cambridge University Press, 2005.

[40] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, "Evaluating the reliability of NAND multiplexing with PRISM," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 10, pp. 1629–1637, 2005.

[41] J. Norris, *Markov Chains*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[42] E. Pavese, V. Braberman, and S. Uchitel, "Automated reliability estimation over partial systematic explorations," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE'13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 602–611.

[43] A. Puggelli, W. Li, A. Sangiovanni-Vincentelli, and S. Seshia, "Polynomial-time verification of PCTL properties of MDPs with convex uncertainties," in *Computer Aided Verification*. Springer, 2013, pp. 527–542.

[44] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[45] G. Rodrigues, D. Rosenblum, and S. Uchitel, "Using scenarios to predict the reliability of concurrent component-based software systems," in *Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering*, ser. FASE'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 111–126.

[46] G. N. Rodrigues, D. S. Rosenblum, and S. Uchitel, "Reliability prediction in model-driven development," in *Proceedings of the 8th international conference on Model Driven Engineering Languages and Systems*, ser. MoDELS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 339–354.

[47] S. Sahni, "Computationally related problems," *SIAM Journal on Computing*, vol. 3, pp. 262–279, 1974.

[48] K. Sen, M. Viswanathan, and G. Agha, "Model-checking Markov chains in the presence of uncertainties," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2006, pp. 394–410.

[49] G. W. Stewart and J. Sun, *Matrix Perturbation Theory*. Academic Press, 1990.

[50] G. Su and D. S. Rosenblum, "Asymptotic bounds for quantitative verification of perturbed probabilistic systems," in *Proceeding of the 15th International Conference on Formal Engineering Methods*, L. Groves and J. Sun, Eds. Springer, 2013.

[51] ——, "Nested reachability approximation for discrete-time markov chains with univariate parameters," in *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings*, 2014, pp. 364–379.

[52] ——, "Perturbation analysis of stochastic systems with empirical distribution parameters," in *Proceeding of the 36th International Conference on Software Engineering (ICSE'14)*, 2014.

[53] M. Y. Vardi, "Automatic verification of probabilistic concurrent finite state programs," in *26th Annual Symposium on Foundations of Computer Science*. IEEE, 1985, pp. 327–338.

PLACE PHOTO HERE

Guoxin Su received a PhD degree in computer science from the University of Technology Sydney, Australia and a bachelor's degree and a master's degree from the Philosophy Department at Sun Yat-sen University, China. He is currently a postdoctoral research fellow with the Department of Computer Science, School of Computing, National University of Singapore, Singapore. His research interests include probabilistic model checking and formal verification of software systems.

PLACE PHOTO HERE

Yuan Feng is a Professor at the Centre for Quantum Computation and Intelligent Systems (QCIS), University of Technology Sydney (UTS), Australia. He received his BS and PhD degrees from the Department of Applied Mathematics and the Department of Computer Science and Technology, Tsinghua University, in 1999 and 2004, respectively. Before joining UTS in 2009, he was an Associate Professor at Tsinghua University. His research interests include the theory of quantum information and quantum computation, quantum programming, and probabilistic systems.

PLACE PHOTO HERE

Taolue Chen received the bachelor's and master's degrees from the Nanjing University, China, both in computer science. He was a junior researcher (OiO) at CWI and acquired PhD degree from the Free University Amsterdam, The Netherlands. He is currently a senior lecturer in the Department of Computer Science, Middlesex University London, UK. Prior to this, he was a research assistant in the University of Oxford, UK, and a postdoctoral researcher in the University of Twente, The Netherlands. His research interests include formal verification and synthesis of stochastic systems, model checking, concurrency theory, process algebra, and computational complexity.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2015.2508444, IEEE Transactions on Software Engineering

PLACE PHOTO HERE

David S. Rosenblum received the PhD degree from Stanford University in 1988. He is a professor in the Department of Computer Science and Dean of the School of Computing at the National University of Singapore (NUS). Before joining NUS, he was a research scientist at AT&T Bell Laboratories in Murray Hill, New Jersey, from 1988 to 1996; an associate professor at the University of California, Irvine, from 1996 to 2002; the chief technology officer and principal architect of PreCache, Inc., from 2001 to 2003; and professor of software systems at University College London from 2004 to 2011. His research interests include probabilistic modeling and analysis, and the design and validation of mobile, context-aware ubiquitous computing systems. He is the recipient of the 2002 ICSE Most Influential Paper Award for his ICSE 1992 paper on assertion checking, and the first ACM SIGSOFT Impact Paper Award in 2008 for his ESEC/FSE 1997 paper on Internet-scale event observation and notification (coauthored with Alexander L. Wolf). He is a Fellow of the ACM and the IEEE, the Past Chair of ACM SIGSOFT, and the Editor-in-Chief of the ACM Transactions on Software Engineering and Methodology.

0098-5589 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.