Verifying the Long-Run Behavior of Probabilistic System Models in the Presence of Uncertainty

Anonymous Author(s)

ABSTRACT

Verifying that a stochastic system is in a certain state when it has reached equilibrium has important applications. For instance, the probabilistic verification of the long-run behavior of a safety-critical system enables assessors to check whether it accepts a human abort-command at any time with a probability that is sufficiently high. The stochastic system is represented as a probabilistic model, a long-run property is asserted and a probabilistic verifier checks the model against the property.

However, existing probabilistic verifiers do not account for the imprecision of the probabilistic parameters in the model. Due to uncertainty, the probability of any state transition may be subject to small perturbations which can have dire consequences for the veracity of the verification result. In reality, the safety-critical system may accept the abort-command with an insufficient probability.

In this paper, we introduce the first probabilistic verification technique that accounts for uncertainty in the verification of long-run properties of a stochastic system. We present a mathematical framework for the asymptotic analysis of the stationary distribution of a discrete-time Markov chain, making no assumptions about the distribution of the perturbations. Concretely, our novel technique computes upper and lower bounds on the long-run probability, given a certain degree of uncertainty about the stochastic system.

1 INTRODUCTION

Probabilistic verification is a powerful and mature technology which aims to verify systems that exhibit stochastic rather than deterministic behavior. The stochastic system is represented by a stochastic model, such as a discrete-time Markov chain (DTMC). A probabilistic model checker verifies this model against a given system property, such as the minimum (or maximum) long-run probability or reachability of a certain state. For instance, the lifecycle of software developed in a large company can be modeled as a DTMC where a state represents the current development stage of a software, and the state transition probabilities can be determined empirically from the life-cycle of software developed in the past. A probabilistic model checker, such as PRISM [14], can then check whether the long-run probability that a software is in the (error-free) deployment stage exceeds some threshold.

However, probabilistic verification does not account for the imprecision of the probabilistic parameters in the model of the stochastic system. Often, the specified transition probabilities are accurate only to some degree. For instance, the transition probabilities in the DTMC model of the company’s development life-cycle are computed as the sample mean over past instances. However, a sample mean is subject to variance and approximates the population mean only with some accuracy. The company may hire new developers, develop different software, or improve the development process. All of these contribute to our uncertainty about the specific probability to transition from one development stage to another.

Previous work has successfully applied perturbation theory to many aspects of probabilistic model checking, including different stochastic formalisms, different kinds of properties to be verified and different measures of perturbation distance, as well as applications of the ideas to self-adaptive software, QoS monitoring and cloud computing [16, 19–22].

This paper presents results on dealing with uncertainty when verifying long-run properties of DTMCs, where we seek to determine the probability that a system—whose stochastic behavior is subject to perturbation—will be in a particular state of interest once the system has reached a steady state, or equilibrium. We propose to model a stochastic system that is subject to uncertainty as a parameterized DTMC where the specific probability in each transition is subject to random perturbations. We introduce a probabilistic verification technique that provides an upper and lower bound on the long-run probability for each state in the parameterized DTMC. The bounds characterize the worst-case consequences of uncertainty on the (long-run) probability that the system is in a certain state at any time after it has reached equilibrium.

The verification of long-run probabilities is a very useful and versatile application of probabilistic model checking and can be applied to problems in software engineering as well as other domains. For instance, it allows us to check whether the probability that a safety-critical system accepts an external abort-command at an arbitrary point in time is sufficiently high. It also allows us to check which functions are most critical for a system because they are executed most often, or which web-pages on a server are important because they are visited with greater likelihood.

We introduce the pertinent concepts and evaluate our novel probabilistic verification technique using three case studies. In the first case study, we discuss the probabilistic verification of a long-run property in the development life-cycle of a large mobile app development company. In the second case study, we investigate the impact of estimating the transition probabilities in the DTMC from empirical data (i.e., using maximum-likelihood estimation). Of course, a sample is subject to some variance resulting in small perturbations for the transition probabilities. The DTMC models web-pages on a server, and the verification task is to check the importance of those web-pages given a sample of click-stream data.
In the third case study, we investigate the impact of changes to the program or its workload on determining the functions that are most critical for the program. The DTMC models the sequence of function calls in the program, and the verification task is to check the probability that a function is exercised at any time after equilibrium, given that the program or workload may slightly change.

The main contributions of this work are as follows:

(1) We introduce the first probabilistic verification technique that accounts for uncertainty in the verification of long-run properties of a stochastic system. Our technique uses perturbation analysis to provide upper and lower bounds on the system’s long-run properties which represent the worst-case consequences of uncertainty.

(2) We present a mathematical framework for the asymptotic analysis of the stationary distribution of a (reducible or irreducible) DTMC when the transition probabilities are subject to a random perturbation. We make no assumptions about the distribution of the perturbations, reflecting the reality that the degree of uncertainty for specific state transitions is often unknown.

(3) We present a prototype implementation of our work in Python. We evaluate our technique on three case studies. Our experiments indicate that our technique is able to provide an accurate estimation of the worst-case consequences of uncertainty on verification of long-run properties in DTMCs.

The remainder of the paper is organized as follows. Section 2 introduces a running example of the life-cycle of software development. An introduction to basic concepts of probabilistic model checking of DTMCs are presented in Section 3. Section 4 explains our technical approach for analyzing the worst-case consequences of the uncertainty on probabilistic verification of long-run properties in DTMCs. Section 5 presents our experimental evaluation and results. Section 6 discusses related work. Finally, Section 7 summarizes our contributions and discusses our future work.

2 MOTIVATING EXAMPLE

2.1 A Markov Chain Model For The Software Development Life-Cycle of Mobile Apps

Throughout the paper, we exemplify the pertinent concepts and approaches based on the software development life-cycle of a large mobile app development company. For this company, each mobile app goes through the following stages:

- **Early Stage** ($s_0$). The app is created and actively developed. Once the app is feature-complete, it goes to incubation ($s_1$). However, at this stage an app’s development can also be abandoned ($s_3$), e.g., for budget reasons.

- **Incubation** ($s_1$). The app implements all the intended functionality but requires some improvements and fixes. If it turns out that more features or substantial improvements are required, the app goes back to the early stage ($s_0$). If the project is discontinued, the app is retired ($s_4$). However, if the app reaches a certain maturity, it goes to deployment ($s_2$). Once it is deployed, it cannot go back to incubation ($s_1$).

- **Deployment** ($s_2$). The (updated) app is uploaded to the app store and customers start to download and use the app. Deployment is part of the main development life-cycle. If a user reports a bug, the app is marked as buggy ($s_5$).

- **Abandoned** ($s_3$). The app development is discontinued at the early stage. Once the development of an app is marked as abandoned, it remains abandoned.

- **Retired** ($s_4$). The app development is discontinued at the incubation stage. Once an app is retired, it remains retired.

- **Repair** ($s_5$). A developer creates a patch for the buggy app and submits it for peer review. If the patch is accepted, the app goes back to deployment ($s_2$), i.e., the fixed version is uploaded to the app store. If the patch is rejected, the app is again marked as buggy ($s_5$).

- **Buggy** ($s_6$). The app contains a bug that was reported and needs to be fixed. The app is send for repair ($s_5$).

To understand and improve the development life-cycle of their mobile apps, the company records the various stages that each app undergoes. For instance, it can empirically determine the proportion of apps in the **early stage** ($s_0$) that reach **incubation** ($s_1$) or that are **abandoned** ($s_3$). Thus, the development life-cycle can be modeled as a discrete-time Markov chain with each development phase as a state. Each progression from phase to phase as **transition** probability, and the proportion of apps progressing from one phase to another as **transition probabilities**. The Markov model for the company’s development life-cycle is shown in Figure 1. For now, we ignore the dashed rectangles and we consider the values of $x_i$ in the transition probabilities to be fixed at $x_1 = \ldots = x_5 = 0$.

2.2 Problem Statement

In this work, we investigate the worst-case consequences of uncertainty on the verification of long-run properties in a discrete-time Markov chain. In the development life-cycle of an app, the company would like to understand the probability that an app, in the long-run, will be in deployment ($s_2$), i.e., graduated from incubation and not buggy. In the second column of Table 1, we can see that this probability $\pi_{s_2} = 0.033$. It is about 25 times more likely that an app is abandoned or retired before it reaches the deployment phase ($\pi_{s_3} + \pi_{s_4} = 0.834$).

However, there are several sources of uncertainty in the modeling. For instance, the recorded life-cycles are only samples that are taken from a larger population; future development life-cycles...
may not be well-represented by the recorded ones. The company may have changed the development process since beginning of the recordings; current life-cycles may not be well-represented by earlier ones. Hence, the empirically determined transition probabilities are accurate only to some degree. What does that mean for computing the probability $\pi_{s_2}$ that an app, in the long run, will be in the deployment stage ($s_2$)?

We model the uncertainty in the transition probabilities as small perturbations in a parameterized Markov chain, where the (symbolic) perturbation vector $\tilde{x}$ is added to the transition probabilities. Figure 1 depicts the Markov chain for our motivating example parameterized with the perturbation vector $\tilde{x} = (x_1, x_2, x_3, x_4, x_5)$.

We do not assume any knowledge about the (distribution of) values for $\tilde{x}$. This reflects the reality that we cannot know the degree of uncertainty for specific state transitions. However, we assume that the total perturbation distance $\delta = \sum_{x_i \in \tilde{x}} |x_i|$ for all parameterized state transitions can be provided (as an upper bound on the total uncertainty in the stochastic process).

Due to uncertainty, the verification of long-run properties in a Markov chain is (1) correct only within certain accuracy bounds, and (2) sensitive to uncertainty in certain states more than in others. Our approach quantifies the accuracy of the computed probability that an app, in the long run, will be in the deployment stage ($s_2$) in the form of asymptotic bounds, called linear perturbation bounds. Our approach will also quantify the sensitivity of the computed probability to uncertainty in each transition in the form of condition numbers.

For our motivating example, the company is interested in the worst-case consequences of uncertainty in the modeling of the development life-cycle if the total uncertainty was at least $\delta = 0.001$, i.e., $|x_1| + |x_2| + |x_3| + |x_4| + |x_5| \leq 0.001$. For this purpose, in the third column of Table 1, we compute a condition number $\kappa_s$ that captures the effect of perturbation vector $\tilde{x}$ in the computation of $\pi_{s_2}$ such that $0 \leq i \leq 6$. As can be seen from Table 1, the greatest condition number in the model of the development life-cycle of a mobile app corresponds to the probability that an app, in the long run, will be retired ($s_4$). In other words, the long run probability of being in $s_4$ is the most sensitive to the perturbation vector $\tilde{x}$. Based on the condition number $\kappa_s$, and $\delta = 0.001$, the last column of Table 1 provides an estimate of the worst-case consequences of uncertainty in the modeling of the development life-cycle calculated as $\pm \kappa_s \delta$ and named as linear perturbation bounds. For instance, the linear perturbation bounds that estimate the worst-case consequences of uncertainty on verification that an app, in the long run, will be in deployment stage ($s_2$) are calculated as $\pi_{s_2} \pm \kappa_s \delta = [0.0329, 0.0331]$.

### 3 BACKGROUND

#### 3.1 Discrete-Time Markov Chains

A discrete-time Markov chain (DTMC) represents the stochastic behavior of a probabilistic system. Specifically, a DTMC is a transition system with probability distributions for the successors of each state. Formally, a DTMC is defined as follows:

**Definition 1.** Discrete-Time Markov Chain (DTMC). A DTMC is a tuple $\mathcal{D} = (S, P, S_{\text{init}}, AP, L)$ where $S$ is a finite set of states, $P : S \times S \to [0, 1]$ is a probabilistic transition function such that $\forall s \in S$. $\sum_{s' \in S} P(s, s') = 1$. $S_{\text{init}} \subseteq S$ is a set of initial states, $AP$ is a finite set of atomic propositions, and $L : S \to 2^{AP}$ a labeling function.

In addition, we define $i_{\text{init}} : S \to [0, 1]$ as the initial state distribution, such that $\sum_{s \in S} i_{\text{init}}(s) = 1$ and $\forall t \notin S_{\text{init}}, i_{\text{init}}(t) = 0$. For instance, in Figure 1 we have $S = \{s_1 | 0 \leq i \leq 6\}$, $S_{\text{init}} = \{s_0\}$, $AP = \{\text{early stage}, \text{incubation}, \text{deployment}, \text{abandoned}, \text{retired}, \text{repair, buggy}\}$, and for instance $L(s_0) = \text{early stage}$ and $P(s_0, s_1) = 0.5$. Furthermore, $i_{\text{init}}(s_0) = 1$ and $i_{\text{init}}(s_i) = 0 \text{ for } i > 0$.

**Definition 2.** Sub-DTMC. Let $\mathcal{D} = (S, P, S_{\text{init}}, AP, L)$ be a DTMC. A sub-DTMC of $\mathcal{D}$ is a tuple $(S', P')$ such that $\emptyset \neq S' \subseteq S$, and $P' : S' \to [0, 1]$ where for all $s, s' \in S'$, $P'(s, s') = P(s, s')$ and $\sum_{s' \in S'} P'(s, s') = 1$.

The digraph of $\mathcal{D}$, denoted as $G(S, E)$, is induced as there is one node for each state $s \in S$ and a directed edge $(s, t) \in E$ if only if $P(s, t) > 0$. An infinite path in a DTMC $\mathcal{D}$ is a sequence of the form $s = s_0 s_1 s_2 \ldots$ such that $s_i \in S$ and $P(s_i, s_{i+1}) > 0$ for all $i \geq 0$. A finite path $\omega$ is a prefix of an infinite path $\pi$ ending in a particular state where $\omega = s_0 s_1 \ldots s_n$ such that $n = |\omega|$. In this context, we say a state $s_n$ is reachable from $s_0$ if there is a finite path $\omega$.

A set of states $S' \subseteq S$ induces a strongly connected subgraph (SCS) of a DTMC $\mathcal{D}$ if only if for all $s, t \in S'$ there is a path from $s$ to $t$ visiting only states from $S'$. A strongly connected component (SCC) of $\mathcal{D}$ is a maximal (w.r.t. $\subseteq$) SCS of $S$. A bottom strongly connected component (BSCC) of $\mathcal{D}$ is an SCC $S'$ from which no state outside $S'$ is reachable. We denote BSCC($\mathcal{D}$) as the set of all BSCCs of the underlying digraph of $\mathcal{D}$.

A DTMC $\mathcal{D}$ is said irreducible if all its states belong to a single BSCC. Irreducibility of a DTMC is important for convergence to equilibrium as $n \to \infty$, because the convergence should be independent of any start state. In case, a DTMC is not irreducible, we say that is reducible. The DTMC in our motivating example is reducible, as the states belong to three BSCCs shown as dashed boxes in Figure 1.

Given a DTMC $\mathcal{D}$, the period of state $s$ is defined as $d(s) = \gcd(n \in \mathbb{N}_+ : P^n(s, s) > 0)$. State $s$ isaperiodic if $d(s) = 1$ and periodic if $d(s) > 1$. In this context, a DTMC $\mathcal{D}$ is said to be aperiodic if all its states are aperiodic. Correspondingly, if all states of $\mathcal{D}$ are periodic then $\mathcal{D}$ is periodic.

<table>
<thead>
<tr>
<th>Stages</th>
<th>$\pi_{s_2} (\delta = 0)$</th>
<th>$\kappa_s$</th>
<th>Linear Bounds</th>
<th>$\pi_{s_2} - \kappa_s \delta$</th>
<th>$\pi_{s_2} + \kappa_s \delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>s0</td>
<td>0.000</td>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>s1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.0000</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>s2</td>
<td>0.033</td>
<td>0.067</td>
<td>0.0329</td>
<td>0.0331</td>
<td></td>
</tr>
<tr>
<td>s3</td>
<td>0.667</td>
<td>0.222</td>
<td>0.6667</td>
<td>0.6672</td>
<td></td>
</tr>
<tr>
<td>s4</td>
<td>0.167</td>
<td>0.333</td>
<td>0.1667</td>
<td>0.1673</td>
<td></td>
</tr>
<tr>
<td>s5</td>
<td>0.067</td>
<td>0.133</td>
<td>0.0668</td>
<td>0.0671</td>
<td></td>
</tr>
<tr>
<td>s6</td>
<td>0.067</td>
<td>0.133</td>
<td>0.0668</td>
<td>0.0671</td>
<td></td>
</tr>
</tbody>
</table>
3.2 Parametric Discrete-Time Markov Chains

Inspired by previous studies [8, 15, 21, 22], we use a parametric model to integrate the uncertainty which is expressed as small perturbations to transition probabilities in the DTMC.

**Definition 3.** Parametric discrete-time Markov chain (PDTMC). Let \( D = (S, P, S_{\text{init}}, AP, L) \) be a DTMC and \( \vec{x} = (x_1, \ldots, x_m) \) be a perturbation vector. A PDTMC is a tuple \( D[\vec{x}] = (S, P[\vec{x}], S_{\text{init}}, AP, L) \) where \( P[\vec{x}] \) is a parametric transition function based on \( P \), and \( S, S_{\text{init}}, AP \) and \( L \) are defined as Definition 1.

Figure 1 depicts a parameterized DTMC where \( \vec{x} = (x_1, x_2, x_3, x_4, x_5) \).

3.3 Long-Run Properties in DTMCs

In probabilistic model checking of DTMCs, long-run, also named as steady-state, properties are used to analyze the reliability of probabilistic systems and to obtain performance parameters such as throughput, delay, loss probability, etc.

Formally, given a DTMC \( D \), the long-run properties study the limit behavior of the probability vector \( \pi^D(t) = [\pi^D_1(t), \ldots, \pi^D_n(t)] \) when time tends to infinity \( (\lim_{t \to \infty} \pi^D(t)) \) [5]. This limit when exists is called steady-state probability vector, and it is written as \( \pi^D = [\pi^D_1, \ldots, \pi^D_n] \). Intuitively, we can interpret \( \pi^D_s \) as the long-run mean fraction of time the DTMC \( D \) is in state \( s \).

In this paper we are interested in the computation of the long-run probability of being in a particular state \( s \) having started in initial states of a given DTMC \( D \). We denote this probability as \( \pi^D_s \). For convenience, we simply mention \( \pi_s \) instead of \( \pi^D_s \) when \( D \) is clear in the context.

To compute long-run probabilities in DTMCs, previous studies [5, 18] have demonstrated that the steady-state probability vector exists if the DTMC is irreducible and aperiodic. Unfortunately, it is not the case in all probabilistic models. Thus, to avoid with periodic considerations, we use the long-run probability for computing the steady-state probability vector \( \pi^D \) of a DTMC \( D \). Consequently, the procedure for computing the long-run probability \( \pi^D_s \) is based on the following conditions:

3.3.1 DTMC \( D \) is irreducible. Let \( D = (S, P, S_{\text{init}}, AP, L) \) be an irreducible DTMC and given a particular state \( s \in S \), we compute the long-run probability \( \pi^D_s \) as follows:

\[
\pi^D_s = \sum_{s' \in S} \left( \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} P^K(s', s) \right),
\]

where \( P^K(s', s) \) denotes the probability of DTMC \( D \) being in state \( s \) after exactly \( k \) transitions when starting in state \( s' \).

3.3.2 DTMC \( D \) is reducible. Let \( D = (S, P, S_{\text{init}}, AP, L) \) be a reducible DTMC and given a particular state \( s \in S \). To compute the long-run probability \( \pi^D_s \), we first identify the set of all BSCCs of the underlying digraph of \( D \), denoted as \( \text{BSCC}(D) \). Then, we evaluate whether state \( s \) belongs to any BSCC \( K \in \text{BSCC}(D) \).

- In the affirmative case, the probability \( \pi^D_s \) is the combination of i) the probability of reaching the BSCC \( K \) having started in the initial states of the model, denoted as \( P_K^K(K) \), and ii) the long-run probability of state \( s \) in a sub-DTMC \( D(K) = (K, P_K) \) of \( D \), induced by BSCC \( K \), where \( P_K = (P(s, t))_{s, t \in K} \). In short,

\[
\pi^D_s = \pi_K^K(K) \cdot \pi^D_s.
\]

Since every BSCC is irreducible, we follow Equation 1 for computing the probability \( \pi^D_s \).

- Otherwise, the long-run probability \( \pi^D_s = 0 \).

4 PERTURBATION APPROACH

In this section, we present our technical approach for estimating the worst-case consequences of uncertainty on the probabilistic verification of long-run properties in DTMCs based on perturbation analysis. We first use a parameterized DTMC to model the presence of uncertainty, which is expressed as small perturbations to transition probabilities. Then, we define a variation function that presents a mathematical characterization of the perturbation in the presence of uncertainty, which is expressed as small perturbations to transition probabilities. In particular, this function captures the difference between the verification of a long-run property against a given DTMC and its associated unperturbed DTMC. Lastly, we look for the linear fragment of the variation function that provides a useful approximate solution of this function. Based on this linear fragment and using the concept of condition numbers, we compute linear perturbation bounds that reveal the worst-case consequences of perturbations to the verification of long-run properties in DTMCs.

4.1 Dealing with Uncertainty

Let \( D = (S, P, S_{\text{init}}, AP, L) \) be the unperturbed DTMC. We first identify the probabilistic transitions that are vulnerable to perturbations. Then, we capture the presence of these perturbations by using a perturbation vector \( \vec{x} = (x_1, \ldots, x_m) \) where \( m \) is the total number of perturbation variables in \( \vec{x} \). Based on the unperturbed transition function \( P \), we incorporate the perturbation vector \( \vec{x} \) into the parametric probabilistic transition \( P[\vec{x}] \), which is defined as follows:

**Definition 4.** Parametric Probabilistic Transition Function. Given a perturbation vector \( \vec{x} = (x_1, \ldots, x_m) \) and a non-parametric probabilistic transition \( P \), \( P[\vec{x}] \) is the parametric probabilistic transition function such that \( \sum_{s, t \in S} P[\vec{x}](s, t) = 1 \) and each entry \( P[\vec{x}](s, t) \) of \( P[\vec{x}] \) is defined as follows:

\[
P[\vec{x}](s, t) = \begin{cases} P(s, t) & \text{if } P(s, t) = 0 \text{ or } P(s, t) = 1 \\ P(s, t) + x_i & \text{if there exists } x_i \in \vec{x} \text{ such that } \text{val}(s, t, x_i) \\ P(s, t) & \text{otherwise} \end{cases}
\]

where \( \text{val}(s, t, x_i) = \text{true if the transition from state } s \text{ to state } t \) is vulnerable to perturbation and \( x_i \in \vec{x} \) represents the perturbation for the transition from \( s \) to \( t \).

Note that a perturbed entry of the form \( P(s, t) + x_i \) is composed by the non-symbolic part \( P(s, t) \), and the perturbation variable \( x_i \in \vec{x} \), which captures the presence of uncertainty.

**Proposition 1.** Given a vector \( \vec{x} = (x_1, \ldots, x_m) \), let \( \mathcal{F} \) be a partition on \( \{1, \ldots, m\} \). A sub-vector \( \vec{y} = (x_j, x_k, \ldots, x_k) \) is an independent perturbed sub-vector in the partition \( \mathcal{F} \) such that \( t \leq j < k \leq m \). For all \( \vec{y} \in \mathcal{F}, \sum_{s \in S} P[\vec{x}](s, t) = \sum_{s \in S} \sum_{i \in E} P(s, t) + \sum_{x_i \in \vec{x}} \sum_{j \in \mathcal{F}} \text{val}(s, t, x_j) \) for some \( s \in S \).

Proposition 1 states that each element of \( \vec{x} \) falls into an independent perturbed sub-vector \( \vec{y} \) and all variables of a sub-vector...
are used. On the other hand, it is always assumed that the vector \( \mathbf{x} = (x_1, \ldots, x_m) \) is within the set \( \mathcal{U} = \{ \mathbf{x} \in \mathbb{R}^m \mid \forall y \in \mathcal{S}, \sum_{x_i \in y} x_1 = 0, P(\mathbf{x}) = P \} \). Note that we say \( P(\mathbf{x}) = P \) when they have exactly the same non-zero entries. In this context, we refer the parametric DTMC \( \mathcal{D}[\mathbf{x}] \) (Definition 3) as the perturbed version of DTMC \( \mathcal{D} \).

To illustrate the definitions above, consider again the DTMC model of the development life-cycle of a mobile app presented in Figure 1. In the following, we denote this DTMC as \( \mathcal{D}^{lc} \). As we introduced in Section 2, there are several sources of uncertainty in the modeling of \( \mathcal{D}^{lc} \), which are expressed as small perturbation in the probabilistic transitions of the model. As Figure 1 shows, the uncertainty in the model is captured by the perturbation vector \( \mathbf{x} = (x_1, x_2, x_3) \). Note that we only incorporated variables into transitions between 0 and 1. The presence of \( \mathbf{x} \) in \( \mathcal{D}^{lc} \) has as a consequence the definition of PMC \( \mathcal{D}^{lc}[\mathbf{x}] \) where the only difference between both models is the parametric function \( P(\mathbf{x}) \).

Based on Proposition 1, each perturbation variable \( x_i \in \mathbf{x} \) falls into two independent perturbed sub-vectors: \( y_1 = (x_1, x_2, x_3) \) and \( y_2 = (x_4, x_5) \). As a result, we have \( \mathcal{S} = \{ y_1, y_2 \} \). Note that each sub-vector is associated to the outgoing transitions of a perturbed state. For example, sub-vectors \( y_1 \) and \( y_2 \) are associated to states \( s_1 \) and \( s_2 \), respectively.

### 4.2 Variation Function for Long-Run Properties in DTMCs

This section presents a mathematical characterization of the perturbation on verification of long-run properties in DTMCs. Recall that a variation function captures the difference between the computation of a long-run probability in a perturbed and unperturbed model. Following the computation of long-run probabilities described in Section 3.3, we present two variation functions: i) for irreducible DTMCs, and ii) for reducible DTMCs.

**Definition 5. Variation Function - Irreducible Case.** Let \( \mathcal{D} = (S, \mathcal{P}, \text{Sim}, \mathcal{A}, \mathcal{P}, L) \) and \( \mathcal{D}[\mathbf{x}] = (S, P(\mathbf{x}), \text{Sim}, \mathcal{A}, \mathcal{P}, L) \) be an irreducible DTMC and its parametric irreducible variant, respectively. A variation function of \( \mathcal{D}[\mathbf{x}] \) against the long-run probability of a particular state \( s \in S \) is \( \sigma : (S, \mathcal{U}) \rightarrow [0, 1] \) defined as follows:

\[
\sigma(s, \mathbf{x}) = \pi_s^\mathcal{D}[\mathbf{x}] - \pi_s^\mathcal{D}.
\]  
(3)

Before presenting the variation function for reducible DTMCs, let us recall that given an unperturbed reducible DTMC \( \mathcal{D} \), the computation of \( \pi_s^\mathcal{D} \neq 0 \) if and only if \( s \in K, K \in \text{BSCC}(\mathcal{D}) \). The value of \( \pi_s^\mathcal{D} \) depends on two components:

1. The reachability probability from initial states to BSCC \( K \), denoted as \( P_r^\mathcal{D}(\mathbf{0}, K) \).
2. The long-run probability of state \( s \) in sub-DTMC \( \mathcal{D}(K) \) if and only if \( s \in \text{BSCC} K \). This probability is denoted as \( \pi_s^\mathcal{D}(K) \).

Under those circumstances, we define the variation function for reducible DTMCs as follows:

**Definition 6. Variation Function - Reducible Case.** Let \( \mathcal{D} = (S, \mathcal{P}, \text{Sim}, \mathcal{A}, \mathcal{P}, L) \) and \( \mathcal{D}[\mathbf{x}] = (S, P(\mathbf{x}), \text{Sim}, \mathcal{A}, \mathcal{P}, L) \) be a reducible DTMC and its parametric reducible variant, respectively. Let \( K \in S \) be a BSCC. A variation function of \( \pi_s^\mathcal{D}(K) \) with respect to the long-run probability of a particular state \( s \in K \) is \( \tau : (S, \mathcal{U}) \rightarrow [0, 1] \) defined as follows:

\[
\tau(s, \mathbf{x}) = \Pr\{D[\mathbf{x}](\mathbf{0}, K) \cdot \pi_s^\mathcal{D}(K) \} - \Pr\{D(\mathbf{0}, K) \cdot \pi_s^\mathcal{D}(K)\},
\]  
(4)

where \( \mathcal{D}(K)[\mathbf{x}] \) and \( \mathcal{D}(K) \) represent the perturbed and unperturbed sub-DTMC of \( \mathcal{D}[\mathbf{x}] \) and \( \mathcal{D} \), respectively.

### 4.3 Linear Perturbation Bounds

The key idea behind perturbation theory is to reduce a hard problem into an infinite sequence of relatively simple ones. We compute the linear perturbation bounds that capture the verification effect of uncertainty in the form of condition numbers. These bounds provide a useful approximate solution to the variation function.

We define the linear perturbation bounds i) for irreducible DTMCs, and ii) for reducible DTMCs. For each case, we first define the linear fragment of its corresponding variation function. Then, we define the condition number based on the linear fragment. Last, we compute the linear perturbation bounds based on the condition number and the total perturbation distance \( \delta > 0 \) of the model.

#### 4.3.1 Irreducible Case

Let \( \sigma(s, \mathbf{x}) \) be the variation function for irreducible DTMCs with respect to the long-run probability of a particular state \( s \) (Definition 5). The linear fragment of the variation function \( \sigma(s, \mathbf{x}) \) is defined as follows:

**Theorem 4.1. Linear Fragment of \( \sigma(s, \mathbf{x}) \) - Irreducible Case.** Let \( \sigma(s, \mathbf{x}) \) be the variation function for irreducible DTMCs. For any \( \mathbf{x} \in \mathcal{U} \), the linear fragment \( \sigma_1(s, \mathbf{x}) \) is formulated as follows:

\[
\sigma_1(s, \mathbf{x}) = \sum_{s' \in S} \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} \sum_{i=j=k-1}^{n} (P(\mathbf{x}) P^i(s', s)) - \sum_{k=1}^{n} P^k(s) + \sum_{k=1}^{n} P^k(s).
\]  
(5)

where \( P = P(\mathbf{x}) - P \).

**Proof.** Let us start by analyzing the variation function \( \sigma(s, \mathbf{x}) = \pi_s^\mathcal{D}[\mathbf{x}] - \pi_s^\mathcal{D} \) (Equation 3). Using the solution of the unperturbed \( \pi_s^\mathcal{D} \) as an approximation to the solution of the perturbed \( \pi_s^\mathcal{D}[\mathbf{x}] \), we can express the variation function \( \sigma(s, \mathbf{x}) \) as follows:

\[
\sigma(s, \mathbf{x}) = \sum_{s' \in S} \lim_{n \to \infty} \frac{1}{n} P(\mathbf{x}) P^i(s', s) - \sum_{k=1}^{n} P^k(s) + \sum_{k=1}^{n} P^k(s).
\]  
(6)

Let \( P = P(\mathbf{x}) - P \) and \( \delta = \sum_{s' \in S} |P(\mathbf{x}) - P| \) be the total perturbation distance. By expanding the term \( P \) of Equation 6, we have:

\[
\sum_{k=1}^{n} P(\mathbf{x}) P^k - \sum_{k=1}^{n} P^k = \sum_{i+j=k-1}^{n} (P(\mathbf{x}) P^i) + O(\delta^2).
\]  
(7)

As we look for the linear fragment of \( \sigma(s, \mathbf{x}) \), we ignore any term of more one copy of \( P = P(\mathbf{x}) \) in Equation 7. As a result, we have:

\[
\sum_{k=1}^{n} P(\mathbf{x}) P^k - \sum_{k=1}^{n} P^k = \sum_{k=1}^{n} (P(\mathbf{x}) P^i) + O(\delta^2).
\]  
(8)

where \( x_i \sim O(\delta) \). Based on previous equation, we conclude linear fragment \( \sigma_1(s, \mathbf{x}) \) as Equation 5.

Based on the above linear fragment \( \sigma_1(s, \mathbf{x}) \), we define the condition number and the linear perturbation bounds for irreducible DTMCs as follows:
Quantifying the Impact of Uncertainty on the Long-Run Probability

**Definition 7.** Condition number - Irreducible Case. Let \( \tilde{h}^T \) denote the transpose of vector \( \tilde{h} \). Let \( \tilde{x} \in \mathcal{U} \) be a perturbation vector. We can write \( \sigma_i (s, \tilde{x}) \) in the form \( \sigma_i (s, \tilde{x}) = \tilde{x} \cdot \tilde{h}^T \) for some constant vector \( \tilde{h} \). Thus, we can define the condition number \( \kappa^\text{irr}_s \) for irreducible DTMCs as follows:

\[
\kappa^\text{irr}_s = \max_{i} \frac{\sigma_i (s, \tilde{h})}{\sigma_i (s, \tilde{h})},
\]

(9)

**Definition 8.** Linear Perturbation Bound - Irreducible Case. Let \( \delta = \sum x_i |x_i| \) be the total perturbation distance. Let \( \pi^{\text{irr}}_s \) and \( \kappa^\text{irr}_s \) be the unperturbed long-run probability and the condition number of a particular state \( s \) in an irreducible DTMC \( D \), respectively. A pair of upper and lower linear perturbation bounds for the variation function \( \sigma(s, \tilde{x}) \) are defined as follows:

\[
f^+_{\text{irr}} = \pi^{\text{irr}}_s + \kappa^\text{irr}_s \delta,
\]

(10)\[
f^-_{\text{irr}} = \pi^{\text{irr}}_s - \kappa^\text{irr}_s \delta.
\]

(11)

As an illustration, consider again PDTMC \( D^4 \{x\} \) shown in Figure 1. As the figure shows, the dashed rectangles represent the set of BSCCs of the model: \( K_1 = \{s_2, s_3, s_4, s_5\}, K_2 = \{s_3\} \) and \( K_3 = \{s_4\} \). Suppose we are interested to estimate the worst-case effect of perturbed sub-vector \( \tilde{y}^2 = (x_4, x_5) \) on the verification of the probability that an app, in the long run, will be in deployment \( s_2 \). Since \( s_2 \in K_1 \), we only focus on BSCC \( K_1 \) that induces a sub-PDTMC \( D^4 \{x\}(K_1) = (K_1, P^s_{\text{red}} | K_1) \) where \( P^s_{\text{red}} | K_1 = (P^4_{\text{red}}(x, t))_{x, t \in K_1} \) and \( \tilde{y}^2 \subseteq \tilde{x} \). It is important to notice that the sub-PDTMC \( D^4 \{x\}(K_1) \) is irreducible.

Under those circumstances, we follow the computation of linear perturbation bounds for irreducible DTMCs. First, we define the variation function \( \sigma(s, \tilde{x}) \) for reducible DTMCs as follows:

\[
f^+_{\text{irr}} = \pi^{\text{irr}}_s + \kappa^\text{irr}_s \delta,
\]

(10)\[
f^-_{\text{irr}} = \pi^{\text{irr}}_s - \kappa^\text{irr}_s \delta.
\]

(11)

By substituting the above expansion into Equation 4, we can rewrite the variation function \( \tau(s, \tilde{x}) \) as follows:

\[
\tau(s, \tilde{x}) = \varpi(D) \cdot (\pi^{D(K)}_s - \pi^{D(K)}_s) + \rho(s, \tilde{x}) \cdot \varpi(D(K)) \tilde{x}.
\]

(14)

It is important to note that the term \( \pi^{D(K)}_s - \pi^{D(K)}_s \) in Equation 14 is the definition of variation function for irreducible DTMCs, denoted as \( \sigma(s, \tilde{x}) \) (Definition 5). Thus, by substituting \( \sigma(s, \tilde{x}) \) into Equation 14, we have:

\[
\tau(s, \tilde{x}) = \varpi(D) \cdot (\pi^{D(K)}_s - \pi^{D(K)}_s) + \rho(s, \tilde{x}) \cdot \varpi(D(K)) \tilde{x}.
\]

(15)

In the above equation, the term \( \pi^{D(K)}_s - \pi^{D(K)}_s \) is defined as:

\[
\pi^{D(K)}_s - \pi^{D(K)}_s = \sum \left( \lim_{n \to \infty} \frac{1}{n} \sum_{k=1}^{n} P^k \right)_{\tilde{x} \to (s', s)}.
\]

(16)

Let \( P^\text{red}_K = P^\text{red}_K \tilde{x} - P_K \) and \( \delta = \sum x_i |x_i| \) be the total perturbation distance. In similar way than the irreducible case, expanding the term \( \sum_{k=1}^{n} P^k \tilde{x}^k \), we have:

\[
\sum_{k=1}^{n} P^k \tilde{x}^k = \sum_{k=1}^{n} (P^\text{red}_K + P_K)^k
\]

(17)

\[
= \sum_{k=1}^{n} P^\text{red}_K + \sum_{k=1}^{n} \sum_{j=0}^{k} (P^\text{red}_K)^j (P^\text{red}_K)^{k-j} + O(\delta^3),
\]

(18)

where \( x_i \sim O(\delta) \). It is important for calculating the linear fragment of \( \tau(s, \tilde{x}) \) to ignore any term of more one copy of \( P^\text{red}_K \) in Equation 17. Combining the above equations and Equation 15, we conclude as linear fragment of \( \tau(s, \tilde{x}) \) as Equation 12.

Based on linear fragment \( \tau(s, \tilde{x}) \), we define the condition number and the linear perturbation bound for reducible DTMCs as follows:

**Definition 9.** Condition Numbers - Reducible Case. Let \( \tilde{h}^T \) denote the transpose of vector \( \tilde{h} \). Let \( \tilde{x} \in \mathcal{U} \) be a perturbation vector. We can write \( \tau(s, \tilde{x}) \) in the form \( \tau(s, \tilde{x}) = \tilde{x} \cdot \tilde{h}^T \) for some constant vector \( \tilde{h} \). Thus, we can define the condition number \( \kappa^\text{red}_s \) for reducible DTMCs as follows:

\[
\kappa^\text{red}_s = \max_{i} \frac{\sigma_i (s, \tilde{h})}{\sigma_i (s, \tilde{h})},
\]

(19)
upper and lower linear perturbation bounds for the variation function \( \tau(s, \bar{x}) \) are defined as follows:

\[
\begin{align*}
    f^+_{\text{red}} &= \pi^{D|K}_s \delta + \kappa^{\text{red}}_{s} \delta, \\
    f^-_{\text{red}} &= \pi^{D|K}_s - \kappa^{\text{red}}_{s} \delta.
\end{align*}
\]  

(20)

(21)

To exemplify the computation of linear perturbation bounds in reducible DTMCs, consider again the reducible PDTMC \( D^{cl}[\bar{x}] \) with perturbation vector \( \bar{x} = (x_1, x_2, x_3, x_4, x_5) \) as defined in Figure 1. Suppose again we are interested to estimate the worst-case effect of \( \bar{x} \) on verification of the long-run probability that an app will be in deployment \( (s_2) \) in the PDTMC \( D(K_1)[\bar{x}] \). Recall the set of BSCCs are \( K_1 = \{s_2, s_5, s_6\} \), \( K_2 = \{s_3\} \) and \( K_3 = \{s_4\} \).

By Definition 6 and given that \( s_2 \in \text{BSCC} K_1 \), we first define \( \tau(s_2, \bar{x}) = Pr.D^{cl}[\bar{x}]((\text{BSCC}) \cdot \sigma_i(s_2, \bar{x}) + \rho_i(\bar{x}) \cdot \pi^{D|K_1}_s(\bar{x}) \) as the variation function, where \( \pi^{D|K_1}_s(\bar{x}) \) denotes the long-run probability that an app will be in deployment \( (s_2) \) in the sub-PDTMC \( D(K_1)[\bar{x}] \) induced by BSCC \( K_1 \).

Following Theorem 4.2, the linear fragment of \( \tau(s_2, \bar{x}) \) is defined as \( \tau(s_2, \bar{x}) = Pr.D^{cl}[\bar{x}]((\text{BSCC}) \cdot \sigma_i(s_2, \bar{x}) + \rho_i(\bar{x}) \cdot \pi^{D|K_1}_s(\bar{x}) \) where \( Pr.D^{cl}[\bar{x}]((\text{BSCC}) \cdot \sigma_i(s_2, \bar{x}) + \rho_i(\bar{x}) \cdot \pi^{D|K_1}_s(\bar{x}) \) = 0.666, \( \sigma_i(s_2, \bar{x}) = 1.666x_4 + 0.166x_5 \) (irreducible case), \( \rho_i(\bar{x}) = 0.111x_1 + 0.666x_2 + 0.111x_3 \). Thus, we have \( \tau(s_2, \bar{x}) = 0.0221x_1 + 0.133x_2 + 0.0648x_4 + 0.0092x_3 \). Consequently, based on Definition 9, the condition number \( \kappa^{\text{red}}_{s} = 0.067 \) (shown in Table 1). And, since we are interested in the worst-case effect of \( \bar{x} \) when the total perturbation distance \( \delta = 0.001 \), we compute the upper bound \( f^+_{\text{red}} = 0.033 + 0.067(0.001) = 0.0331 \), and the lower bound \( f^-_{\text{red}} = 0.033 - 0.067(0.001) = 0.0329 \).

### 5 EXPERIMENTAL EVALUATION

In this section, we evaluate the applicability of our approach in two case studies. Our purpose is to demonstrate that the computed asymptotic bounds provide an accurate estimation of the worst-case consequences of uncertainty on verification of long-run behavior in systems modeled as DTMCs. For this purpose, we have developed a prototype implementation in Python.

Figure 2 describes a general overview of our prototype. As can be seen from the figure, for each case study, we first build the system model as a DTMC. Second, based on the designer expertise, we identify transitions vulnerable to perturbations. A perturbation variable is attached to each probabilistic transition. As a result, we obtain the perturbation vector \( \bar{x} \). Third, we formulate the long-run probability of being in a particular state \( s \) to be analyzed, denoted as \( \pi_s \). These mentioned components are the input of our prototype. By following our perturbation approach described in Section 4, our prototype computes a condition number \( \kappa_s \) for \( s \) that captures the effect of uncertainty in the verification of \( \pi_s \). Lastly, our prototype calculates linear perturbation bounds based on the computed condition number and a given perturbation distance \( \delta > 0 \). All our experiments have been conducted on a machine with 3.06 GHz Intel Core Duo processors and 8 GB RAM.

#### 5.1 Clickstream Data

**Model.** As a first case study, we model the stochastic process of users clicking through the web-pages on a server as an *irreducible* DTMC [2, 17]. Specifically, we use public clickstream data collected for a popular news site *msnbc.com* [10]. The data spans the entire day of September 28, 1999 and it contains 989818 logs. Each log represents a clickstream. A clickstream is a sequence of web page categories visited by a given user. The categories are listed in Table 2. The DTMC \( D^{cl} = (S, P, S_{init}, AP, L) \) consists of 17 states \( S = \{s_1, \ldots, s_{17}\} \) which represent the categories of the news site. The set of initial states \( S_{init} \) is composed by the front-page \( (s_1) \). The transitions in \( D^{cl} \) represent the sequences of clicks from one category page to another. \( P(s_i, s_j) \) is calculated by counting all instances of state \( s_i \) that precede state \( s_j \) across all clickstreams. The model contains 17 states and 289 transitions.

We investigate the impact of estimating the DTMC’s transition probabilities from empirical data using maximum-likelihood estimation. The clickstream sample is subject to some variance resulting in small perturbations to the estimated transition probabilities. We are interested in the probability that a user is visiting a certain web-page in the long run. Thus, we calculate the long-run probabilities \( \pi^{D|K}_s \) of all states \( s_i \) with \( 1 \leq i \leq 17 \). In the following, we simply refer to \( \pi^{D|K}_s \) as \( \pi_s \).

**Uncertainty.** The DTMC \( D^{cl} \) that we created from the observed clickstream data is subject to sampling error on each state transition; the sampling error will compound and threaten the validity of any verification results. The clickstream data was collected on a single day. Thus, the data is merely a *sample* of the population of visitors to *msnbc.com*. Sample statistics represent population statistics only to some degree of accuracy. The DTMC’s transition probabilities thus estimated are hence subject to perturbations.

To account for the sampling error, we perturbed each probabilistic transition in the Markov model \( D^{cl} \). In total, we added 289 perturbation variables, \( \bar{x} = (x_1, \ldots, x_{289}) \), for each transition. After perturbing the model, we are interested in analyzing which state is affected most by the perturbation, i.e. which state is most susceptible to compounding sampling errors. For this reason, for each state \( s_i \), we compute its corresponding condition number \( \kappa_s \) by using our approach. Note that since the DTMC \( D^{cl} \) is irreducible, we follow Definition 7 for computing \( \kappa_s \).

We present experimental results for this case study in Table 2. The first two columns depict the set of 17 web page categories, which are ordered from the most visited to least visited. The next column shows the probability \( \pi_{s_i} \) that a visitor, in the long-run, visits page \( s_i \), given the ideal scenario \( (\delta = 0) \). The last column depicts the corresponding condition number \( \kappa_{s_i} \).

As can be seen from Table 2, in an unperturbed model, the most visited page is the front-page (15.79%), followed by BBS (10.99%)
and news (10.19%). On the contrary, the least visited pages are msn-news (0.42%) and msn-sport (0.84%). In Figure 3 we compare the long-run probabilities in the unperturbed model (x-axis) with the condition numbers for each state $s_i$. The condition numbers indicate the effect of perturbing vector $\pi$ during the computation of $\pi_{s_i}$, i.e. the impact of the unknown sampling error on the calculated long-run probabilities.

**Observations.** Figure 3 allows us to make several insightful observations. First, it reveals that the msn-news page (denoted as $s_{16}$)—even though it is one of the least visited pages with $\pi_{s_{16}} = 0.0084$—is very sensitive with respect to perturbations of the transitions probabilities. We can draw this conclusion from the large condition number, $\kappa_{s_{16}} = 13.899$. This finding indicates that small changes in the transition probabilities may have a severe impact on the number of visits to the msn-news page. Second, the sensitivity of the front-page, which is the most visited page, has a condition number ($\kappa_{s_{1}} = 4.0537$) that is close to the median of 4.2728. Thus, the front-page is not particularly susceptible to perturbations of transition probabilities. Finally, inspecting Figure 3 we cannot find a clear correlation between the long-run probability and the condition number. This implies that the effect of uncertainty is undiscriminating for the most and least visited web-pages.

**Soundness.** To evaluate whether our perturbation techniques provide a sound estimate of the worst-case consequences of uncertainty in the computation of long-run probabilities, we introduce

$$\pi_s = \frac{\pi_0}{\sum \pi_0}$$

where $\pi_0$ is the initial probability vector and $\pi_s$ is the probability vector at time $s$. This implies that the effect of uncertainty is undiscriminating for the most and least visited web-pages.

**Table 2: Experimental Results for Clickstream Data**

<table>
<thead>
<tr>
<th>Categories</th>
<th>$\pi_{s_i}$</th>
<th>$\kappa_{s_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front-page</td>
<td>0.1579</td>
<td>4.0537</td>
</tr>
<tr>
<td>BBS</td>
<td>0.1099</td>
<td>6.9263</td>
</tr>
<tr>
<td>News</td>
<td>0.1019</td>
<td>3.0490</td>
</tr>
<tr>
<td>Weather</td>
<td>0.0979</td>
<td>12.146</td>
</tr>
<tr>
<td>Local</td>
<td>0.0966</td>
<td>5.5212</td>
</tr>
<tr>
<td>Misc</td>
<td>0.0783</td>
<td>4.3953</td>
</tr>
<tr>
<td>On-air</td>
<td>0.0608</td>
<td>3.3277</td>
</tr>
<tr>
<td>Sports</td>
<td>0.0549</td>
<td>4.2728</td>
</tr>
<tr>
<td>Opinion</td>
<td>0.0504</td>
<td>8.7741</td>
</tr>
</tbody>
</table>

**Table 3: Accuracy of the Linear Perturbation Bounds**

<table>
<thead>
<tr>
<th>Categories</th>
<th>$\kappa_{s_i}$</th>
<th>$\delta$</th>
<th>$\Delta \pi_{s_i}$</th>
<th>$\pm \kappa_{s_i} \delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front-page</td>
<td>4.0537</td>
<td>0.1579</td>
<td>0.1368</td>
<td>-0.0211</td>
</tr>
<tr>
<td>Msn-news</td>
<td>13.899</td>
<td>0.0084</td>
<td>0.0045</td>
<td>-0.0039</td>
</tr>
<tr>
<td>Weather</td>
<td>12.146</td>
<td>0.0979</td>
<td>0.0587</td>
<td>-0.0392</td>
</tr>
<tr>
<td>Opinion</td>
<td>8.774</td>
<td>0.0504</td>
<td>0.0337</td>
<td>-0.0167</td>
</tr>
</tbody>
</table>

Table 3. As the table shows, we have restricted our evaluation to the most visited page (front-page) in the unperturbed scenario and three of the most sensitivity pages. To provide some intuition about the worst perturbation that might affect the model, we compute $\delta = 21.359$ as the 90% bootstrap confidence interval with around 10% clickstream data as re-sample size. In other words, 0.0726 perturbation for each transition approximately. Given that the greatest probabilities are located in $P(s_i, s_j)$, these probabilities have been perturbed $–0.0726$. Then, $\sum P(s_i, s_j) \approx 0.0726, i \neq j$ as a consequence of the sum of the probabilities of transitions has to normalize to 1. In short, the total distance of the perturbed model is $\delta^* = 0.1452$.

As can be seen from Table 3, for each page, we calculate the long-run probability against a non-parametric DTMC with distance $\delta^*$. Then, we compare the perturbed and unperturbed long-run probability, denoted as $\Delta \pi_{s_i}$, against our estimate $\pm \kappa_{s_i} \delta$ of the worst-case effect of uncertainty in the computation of $\pi_{s_i}$. Our experimental results show that, for each table, the estimate $\pm \kappa_{s_i} \delta$ provide a safe prediction of the impact of perturbations. For example, the difference between the actual probabilities for the perturbed and unperturbed models for the front-page ($s_1$) is $-0.0211$, which lies within our estimation of $\pm 0.5545$.

In summary, using perturbation analysis we succeed in detecting a web page category (msn-news) that is very sensitive to small changes to the transition probabilities. Even minor changes to the transition probabilities might significantly impact the frequency of visits to the msn-news pages. Perturbation analysis of discrete-time Markov chain reflects uncertainties in the model. These uncertainties include (but are not limited to) uncertainties due to sampling error and changing user behavior. The insights generated by our perturbation analysis will be very helpful for capacity planning and related activities.

**5.2 Call Invocation Sequences**

In the previous section we analyzed clickstream data to evaluate how the proposed perturbation technique can help to predict web server traffic and cope with uncertainties in the analyzed model. In this section we continue by investigating how the software engineering community can benefit from our approach, in particular how the technique can help during software maintenance.

**Model.** To this end we model the invocation sequence graph for wc, a program in the coreutils 8.25 tool set. wc parses the standard input or alternatively a set of files and counts various attributes of the read data. By default it counts number of lines, words, and bytes. We focus on wc for a multitude of reasons. First, coreutils is widely distributed and used daily across a diverse user population. Second, wc has a long development history dating
back to at least 1985 making it a very mature software artifact. Third, since coreutils is very mature, we expect a mature test suite which we utilize to collect transition probabilities to model a DTMC and exemplify the strength of the proposed perturbation technique. Finally, we picked the wc program from the coreutils tool set due to prior exposure to its source code. Thus, we were able to manually verify our results using existing prior knowledge.

Similar to the previous case study, we model wc as an irreducible DTMC $\mathcal{D}_{\text{wc}} = (S, P, S_{init}, AP, L)$. The set of states $S$ contains all unique functions called during runtime of wc. Each called function $f_j$ is represented as a state $s_j \in S$. Additionally, $S$ contains two pseudo states, “start” and “end”, that represent program start and program termination, respectively. The set of initial states $S_{init}$ consists of the “start” state. The transitions in $\mathcal{D}_{\text{wc}}$ represent the sequences of calls observed during runtime. We do not model function returns. In particular, $P(s_i, s_j)$ is calculated by counting the fraction of calls to the function $f_j$ represented by $s_j$ that is directly followed by a call to the function $f_i$ represented by state $s_i$. A transition $f_i \rightarrow f_j$ indicates that function $f_j$ might be called either by $f_i$ directly or by any caller of $f_i$. Thus, we model function call sequences not call graphs.

To gather runtime data we instrument wc with LLVM XRay 5.0.1 [1]. In particular, we use LLVM XRay’s capability to collect all function enter and exit events that occurs during runtime. Next, we execute the test suite of coreutils against the instrumented wc binary. We obtain 42 trace files that contain 55,072 logged function calls. We use these 55,072 function calls to create $\mathcal{D}_{\text{wc}}$ as described above. The resulting model contains 46 states representing functions plus 2 pseudo states and 72 transitions.

**Uncertainty.** A common challenge in the context of software maintenance is to judge about the impact of a prospective change. The proposed technique can be used to analyze the sensitivity of a function with respect to call frequency if transition probabilities are perturbed. This can be helpful in the context of software maintenance to, for example, judge about the risk involved in a code change.

For example, let us assume a change is pending. In order to judge about the risk involved in applying the change to the software, especially with regard to performance, we would like to know how often the changed function $f_j$ is expected to be executed. That is, we are interested in computing the long-run probability $\pi(s_i)$. If the function $f_j$ has a high probability of being executed, then any change to this function might have a severe impact on availability and performance of the software artifact. In contrast, if the execution probability is low, then the risk involved is potentially lower. We use the previously described model capturing call invocation sequences to analyze the execution frequencies for wc. To account for uncertainties, we perturb all 40 probabilistic transition, that is $\delta < P(s_i, s_j) < 1$, of the model where $\bar{\mathbf{x}} = (x_1, \ldots, x_{40})$. The experimental results for this case study are shown in Table 4.

The rightmost column in Table 4 contains the condition number (CN), denoted as $k_{s_j}$. Due to space constraints, we present only a limited sub-set of functions of wc. In particular, we show all functions for which the condition number $k_{s_j} > 10$. The functions are ordered by decreasing condition number. We also show the long-run probability $\pi(s_i)$ under no perturbation ($\delta = 0$).

<table>
<thead>
<tr>
<th>Function</th>
<th>$\pi(s_i)^{\text{unc}} (\delta = 0)$</th>
<th>$k_{s_j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_2$</td>
<td>umaxtostr</td>
<td>0.32782</td>
</tr>
<tr>
<td>$s_{11}$</td>
<td>to_uchar</td>
<td>0.00136</td>
</tr>
<tr>
<td>$s_3$</td>
<td>argv_iter</td>
<td>0.10969</td>
</tr>
<tr>
<td>$s_8$</td>
<td>wc_file</td>
<td>0.10931</td>
</tr>
<tr>
<td>$s_7$</td>
<td>wc</td>
<td>0.10930</td>
</tr>
<tr>
<td>$s_6$</td>
<td>fدارة</td>
<td>0.10930</td>
</tr>
<tr>
<td>$s_4$</td>
<td>safe_read</td>
<td>0.10964</td>
</tr>
<tr>
<td>$s_5$</td>
<td>write_counts</td>
<td>0.10946</td>
</tr>
</tbody>
</table>

**Observations.** The first entry in Table 4 describes umaxtostr. Of all the functions observed during runtime umaxtostr is most sensitive to perturbations ($k_{s_2} = 42.7632$). It is also the most often executed function (with a long-run execution probability of 32.789%). Thus, any changes to umaxtostr should be associated with a higher risk. The next entry describes to_uchar. Looking at the execution probability of to_uchar (0.136%) we would predict that changes to this function are likely unproblematic as it is executed very seldom. However, the condition number for to_uchar is relatively high ($k_{s_{11}} = 14.5389$). This indicates that small changes to the transition probabilities can have a severe impact on the execution frequency of to_uchar.

We investigate the call sites of to_uchar. All recorded calls to to_uchar originate from wc.c line 496 and 499. These lines are executed only if wc was instructed to count characters or words, or extract the maximum line length. Thus, how often to_uchar is executed depends strongly on the requested operation. Even though to_uchar is executed only seldom in our test environment any changes to it should be marked as risky since the selection of runtime options might severely increase its invocation rate. The high condition number for to_uchar correctly identifies this sensitivity.

Figure 4 shows the relationship between unperturbed long-run probability (log-scale) and condition number for wc. Similar to Figure 3 in the previous clickstream data scenario we cannot find a clear correlation between the long-run probability and the condition number.

**Soundness.** To evaluate the soundness of our perturbation estimate generated by our technique, we conducted additional experiments, depicted in Table 5. We first built a non-parametric...
model with small perturbation distances ($\delta = 1, 3, 5 \times 10^{-3}$). Second, we compute the long-run probability $\pi_{s_i}$ for a sub-set of functions of wc in the perturbed model, denoted as $\tilde{\pi}_{s_i}$. In particular, we perturbed all probabilistic transitions in the strongly connected subgraph that performs the actual work of counting the requested properties from the input files. That is, we perturb all outgoing transitions from umaxtostr, wc, safe_read, to_uchar, argv_iter, argt_iter_n_args, and call_freefun. Then, we compare the perturbed and unperturbed long-run probabilities, computed as $\Delta \pi_{s_i} = \pi_{s_i} - \tilde{\pi}_{s_i}$. Lastly, we calculate the estimate $\pm \kappa_{s_i} \delta$, which provides the worst-case effect of uncertainty in the computation of $\pi_{s_i}$. For each function the estimate $\pm \kappa_{s_i} \delta$ in Table 5 represents a safe prediction of the worst-case consequences caused by the perturbation. For example, given that $\delta = 1 \times 10^{-3}$, the difference between the actual long-run probabilities of function to_uchar for the perturbed and unperturbed models is $0.138 - 0.136 = 0.002$ with lies within the estimation of $\pm 1.454$. The accuracy of this prediction is replicated where $\delta = 3.5 \times 10^{-3}$. We obtained similar results for all functions presented in Table 5. Finally, note that to_uchar is affected substantially by the perturbations from a relative perspective. In contrast, the long-run probabilities of the other functions are quite stable across the different perturbation distances.

Table 5: Additional Experimental Results for wc

<table>
<thead>
<tr>
<th>Functions</th>
<th>$\pi_{s_1}$</th>
<th>$\pi_{s_2}$</th>
<th>$\pi_{s_3}$</th>
<th>$\pi_{s_4}$</th>
<th>$\pi_{s_5}$</th>
<th>$\pi_{s_6}$</th>
<th>$\pi_{s_7}$</th>
<th>$\pi_{s_8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>wc_file</td>
<td>0.20</td>
<td>0.10</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>argv_iter</td>
<td>10.97</td>
<td>10.9580</td>
<td>10.82</td>
<td>10.71</td>
<td>10.73</td>
<td>10.69</td>
<td>10.67</td>
<td>10.65</td>
</tr>
<tr>
<td>to_uchar</td>
<td>0.136</td>
<td>0.138</td>
<td>0.002</td>
<td>0.138</td>
<td>0.136</td>
<td>0.002</td>
<td>0.138</td>
<td>0.136</td>
</tr>
<tr>
<td>safe_read</td>
<td>10.97</td>
<td>10.9580</td>
<td>10.82</td>
<td>10.71</td>
<td>10.73</td>
<td>10.69</td>
<td>10.67</td>
<td>10.65</td>
</tr>
<tr>
<td>call_freefun</td>
<td>10.93</td>
<td>0.107852</td>
<td>10.77</td>
<td>0.1078</td>
<td>10.66</td>
<td>0.1078</td>
<td>10.59</td>
<td>0.1078</td>
</tr>
</tbody>
</table>

6 RELATED WORK

There is a rich body of literature that investigates finite irreducible Markov chains under perturbation. Many of those works focus on the effect of the perturbed transition matrix of a Markov chain on its steady-state distribution. Similar analysis has also been applied to queueing models [6]. One common technique is the derivation of norm-based bounds for irreducible Markov chains. Specifically, they computed a condition number $\kappa$ for measuring the sensitivity of the Markov chain. Let $E = (E_i) \in \mathbb{R}^{n \times n}$ denote the perturbation. Let $P$ and $P' = P + E$ be probability transition matrices of irreducible Markov chains with respective stationary probability vectors $\pi$ and $\pi'$ satisfying $\pi P = \pi'$, $\pi P = \pi'$, and $\sum_i \pi_i = 1 = \sum_i \pi'_i$. Then, $||\pi - \pi'|| \leq \kappa \cdot ||E||$ for suitable vector and matrix norms.

In those approach, the condition number $\kappa$ has been calculated by using the fundamental matrix of a Markov chain as introduced by Kemeny et al. [11] and Schweitzer [12], the group inverse of $A = I - P$ as presented in [9, 13], the ergodic coefficients [7] and the mean first passage times [3]. All these approaches for computing the condition number were collected and compared by Cho and Meyer [4]. Based on their review, the smallest condition number is computed by using relevant information of the group inverse of $A$. This approach was proposed by Haviv et al. [9] and Kirkland et al. [13].

However, Cho and Meyer also revealed that the computation of the fundamental matrix and the group inverse are usually expensive. For this reason, the condition number in terms of the mean first passage is more feasible to be computed. Likewise, it provides an equivalent condition number to Haviv’s definition based on the underlying structure of the Markov chain.

Together these studies provide valuable insights into the investigation of the sensitivity of the stationary distribution of Markov chains under perturbation. However, these previous studies compute the condition number $\kappa$ based on the perturbation $E$, which is a matrix of prior defined values. Even though we are interested in analyzing the effect of the perturbation in the long-run behavior of MCs, our proposal is different from these previous studies in two points. First, we ignore the defined values that the perturbation might take. Thus, we propose to capture the perturbation as symbolic variables. And, to provide an estimation of the worst-case consequences of the uncertain phenomena. Second, previous studies have not been applied in probabilistic model checking. To the best of our knowledge, no previous publication in the literature has addressed this problem using perturbation analysis.

7 CONCLUSIONS

The main goal of this paper is to estimate the worst-case consequences of uncertainty on verification of long-run properties in DTMCs. Our main contribution is a mathematical framework for asymptotic analysis in the presence of small perturbations to model probabilities on verification of long-run properties against reducible and irreducible Markov chains.

To evaluate our perturbation approach, we implemented a prototype in Python. We ran our experiments on clickstream data and call invocation sequences. These case studies have been modeled as an MC and affected by small perturbations for our evaluation purposes. For each case study, we have evaluated several long-run properties under the presence of perturbations. Our experimental results reveal the importance of our approach at estimating the effect to these perturbations and they provide crucial information for identifying the sensitivity of the long-run properties under uncertainty. This work is the first study on dealing the uncertain phenomena in the probabilistic verification of long-run properties in DTMCs by using perturbation analysis.

There are several directions for further study. For instance, backward analysis that provides the maximum permitted perturbations based on a range of verification results.

10
REFERENCES


