Verification of Variability-Intensive Stochastic Systems with Statistical Model Checking

Sami Lazreg[1], Maxime Cordy[1](✉), and Axel Legay[2]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
{sami.lazreg,maxime.cordy}@uni.lu
[2] UCLouvain, Louvain-La-Neuve, Belgium
axel.legay@uclouvain.be

Abstract. We propose a simulation-based approach to verify Variability-Intensive Systems (VISs) with stochastic behaviour. Given an LTL formula and a model of the VIS behaviour, our method estimates the probability for each variant to satisfy the formula. This allows us to learn the products of the VIS for which the probability stands above a certain threshold. To achieve this, our method samples VIS executions from all variants at once and keeps track of the occurrence probability of these executions in any given variant. The efficiency of this algorithm relies on Algebraic Decision Diagram (ADD), a dedicated data structure that enables orthogonal treatment of variability, stochasticity and property satisfaction. We implemented our approach as an extension of the ProVeLines model checker. Our experiments validate that our method can produce accurate estimations of the probability for the variants to satisfy the given properties.

Keywords: Software product lines · Variability · Statistical Model Checking · Markov chains · Stochastic systems

1 Introduction

When deployed in the field, the correct behaviour of software systems is often put at risk because of unpredictability in the environment (e.g., users or natural phenomena) these systems interact with. That is, the state evolution of the environment is stochastic and this, in turn, entails random non-determinism in the system behaviour. In face of this stochasticity, engineers must provide confidence that the system they build will behave correctly in various situations they cannot control. The difficulty of this task vastly increases when the same engineers develop not a standalone system, but a *Variability-Intensive System (VIS)*.

VISs, such as software product lines [11] and configurable systems [26,28], are systems that one can derive into multiple variants (or configurations). The term variability refers to all the ways in which the variants can differ. In software

T. Margaria and B. Steffen (Eds.): ISoLA 2022, LNCS 13703, pp. 448–471, 2022.
https://doi.org/10.1007/978-3-031-19759-8_27

product lines, such variation points are usually named *features.* Therefore, system variants are uniquely identified by their set of features. Variability makes development activities inherently harder for VISs than for single system development. This is due to the necessity of handling features and their effects throughout all development steps, including verification and validation. Therefore, quality assurance techniques must ensure that *all* system variants that will run in the fieldwork correctly.

VIS variants share many common behaviors and that differ in identified functionalities. For a set of n functionalities one can at worst create 2^n different systems. Checking each system individually would introduce an explosion of time. To overcome these problems, researchers have proposed compact product line representations. These representations make it possible to check all the products in one pass. For nearly 10 years, these approaches were limited to purely Boolean systems. Recently, we have extended the approach to stochastic systems. In this case, we must calculate the probability that a product satisfies the property. This calculation is generally done by extending classical exhaustive algorithms such as those implemented in PRISM.

The variability of VISs and their stochasticity call for dedicated techniques to estimate the *probability* that *any* VIS variant satisfies intended requirements over its behaviour. Engineers should be able to quickly answer questions like *"what is the probability that all variants satisfy a given requirement"*, *"which variants satisfy a given requirement with a desired degree of confidence"*, or *"which variants are the most likely to satisfy a given set of requirements"*. One straightforward way to answer such questions is to apply classical quality assurance techniques to each variant separately to derive an accurate ranking of the variants' likeliness to comply with a requirement. However, getting an accurate answer for all variants may prove difficult, time-consuming and, in turn, even falsify the ranking of these variants with respect to their probability to satisfy the requirements.

In this paper, we propose a method to learn the probability that VIS variants satisfy a given property. Compared to state-of-the-art methods, our approach (a) allows engineers to explicitly model the stochastic distribution of environment events and (b) can effectively assess probabilistic properties across multiple variants. As a side effect, our approach is able to learn the variants of the VIS for which the probability to satisfy the property stays above a given threshold. To achieve this, we lean on Statistical Model Checking (SMC) – a type of verification algorithms that relies on execution sampling and statistical tests to assess model properties [21,22,34]. Statistical model checking consists of learning the probability that the execution of a system will satisfy a given property. The approach elegantly combines (1) a simulation-based algorithm for learning the probability distribution of satisfying the property by observing a fixed number of its executions with (2) runtime verification algorithms applied on these executions. Those runtime verification algorithms naturally depend on the nature of the property to be validated. We develop a novel SMC algorithm that is *family-based*, i.e. it can sample executions from all variants at once *and* keep track of the occurrence probability of these executions in any given variant. The effectiveness of this algorithm relies on Algebraic Decision Diagram (ADD) [2], a dedicated data structure that enables an orthogonal treatment of variability, stochasticity and property satisfaction.

We conduct a preliminary validation of our approach based on case studies from the literature. Our results confirm that our family-based approach produces reliable estimations of the probability for the variants to satisfy given properties. We discuss the factors that influence the effectiveness of our method – i.e., its capability to compute estimations that preserve the differences between the variants – compared to alternatives that analyze each variant separately.

2 Background

2.1 Markov Chains and Variability

We model stochastic system behaviours into Discrete-Time Markov Chains (DTMCs). In such models, (1) the state space S of the system is countable, (2) time elapses at discrete steps and (3) the transitions between states $T \subseteq S \times S$ are stochastic. Hence, one can see DTMCs as a Kripke structure where each transition between two states has a probability to occur at each discrete time step. These probabilities are defined such that they satisfy the usual probability axioms. By Markov's property, the probability of occurrence of a transition depends only on the current state and not on the previously executed transitions. Therefore, the probability for the DTMC to follow a k-length path $\rho = s_0 \dots s_{k-1}$ is equal to the product of the state transition probabilities.

Rodrigues et al. [27] have extended DTMC with variability. The resulting formalism – named *Featured DTMC* (FDTMC) – associates each transition $t \in S \times S$ of the Markov chain with a *probability profile* Π_t that encodes the probability for each variant v to execute t. Such profiles list the set of variants that are following the transitions as well as the probability to take such a transition for a given variant. Precisely, given a set V of variants, Π_t is a function from V to $[0, 1]$. For any $t = (s, s')$ and $v \in V$, $\Pi_t(v) = 0$ means that the variant v cannot execute t, whereas $\Pi_t(v) = 1$ means that, when in state s, v surely executes t at the next discrete time step. For an FDTMC to be consistent, for any state $s \in S$ the probability profile associated to the transitions leaving s must satisfy the probability axioms for all variants. That is, for any $v \in V$, $s \in S$, we have $\sum_{t \in \{(s,s') \in T\}} \Pi_t(v) = 1$. A variant v is typically represented as a set of features (aka variation points) such as $v \in \mathbb{B}^F$.

The product of two probability profiles Π_t and $\Pi_{t'}$ is defined as $(\Pi_t \otimes \Pi_t)(v) = \Pi_t(v)\Pi_{t'}(v)$. The sum $\oplus$ and the division $/$ of two probability profiles are defined similarly. We denote by $\mathbf{0}$ (resp $\mathbf{1}$) the *fixed* profile that associates a value i to every variant $(\forall v \in V, \Pi_{(}v) = i)$. Then for 0 (resp. 1) a complement of Π_t is defined as $(-\Pi_t)$, with $(-\Pi)(v) = 1 - \Pi(v)$, and we also note it $\mathbf{1} - \Pi$.

Based on probability profiles, we define an FDTMC as a tuple (S, ν, V, Π) where S is a countable, non-empty set of states; ν is a vector of size $|S|$ that records the initial probability distribution of every state; V is the set of variants;

$\Pi : (S \times S) \rightarrow (V \rightarrow [0,1])$ is the transition probability function, which assigns a probability profile to each transition. The fact that variant v cannot execute a transition from s to s' is encoded as $\Pi(s,s')(v) = 0$. Note that the probability that any variant executes a k-length path $\rho = s_0, s_1, \ldots, s_k$ in the FDTMC is given by $\Pi_{(s_0,s_1)} \otimes \cdots \otimes \Pi_{(s_{k-1},s_k)}$.

An FDTMC is a concise representation for a set of DTMCs, that is, one per valid variant (or product). The DTMC modelling a particular variant v is obtained by *projecting* the probability profile of each transition onto v. The transition probability function of the resulting DTMC is defined as $P : S \times S \rightarrow [0,1] : P(s,s') = \Pi(s,s')(v)$.

We provide an example of FDTMC in Fig. 1 and a projection of this FDTMC in Fig. 2. This model considers two exclusive features I and J and an optional feature K. The 4 resulting variants can then be expressed in the following feature combinations $\{\{I\}, \{I, K\}, \{J\}, \{J, K\}\}$.

Each transition has either a *fixed* probability value (e.g. the Markov chain transits from s_0 to s_1 with a *fixed* probability profile of **0.5**) or a profile that depends on the variant features. For example, from state s_0, all variants can transit to s_1 (or s_2) with probability 0.5. All variants can also transit from s_1 to state s_0 but with different probabilities (i.e., 0.5 for variants with feature I, 0.2 for ones with J).

Furthermore, only some variant can execute some transitions. For example, only variants with feature J can loop over s_2 while only variants with feature I can loop over s_1. Having a variant that cannot execute a transition t is similar to $\Pi_t(v) = 0$. Similarly, a variant with a probability of 1 to execute a transition is equivalent to a non-stochastic transition (e.g., s_4 to s_3 for variants without K feature).

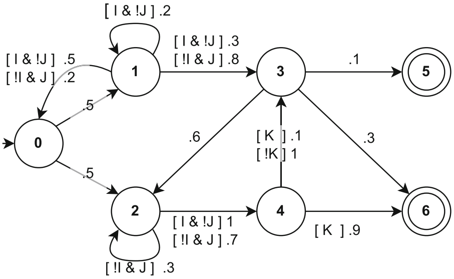


Fig. 1. An illustrative example of stochastic VIS represented as a FDTMC.

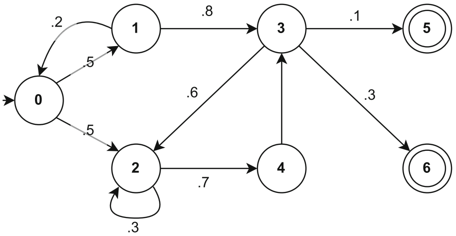


Fig. 2. DTMC resulting from the $FDTMC_{|\{J \wedge \neg I \wedge \neg K\}}$ projection.

2.2 Probabilistic Linear Temporal Logic

We formulate requirements over stochastic systems in the Linear-Time Logic (LTL) [25]. We form LTL formulae according to the following grammar:

$$\phi ::= \top \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \bigcirc\phi \mid \phi \;\mathtt{U}\; \phi$$

where a is an atomic state property; $\bigcirc$ is the next operator; and U is the until operator. From the until operator, one can derive $\Diamond\phi$, which means that the system must eventually reach a state that satisfies ϕ; and $\Box\phi$, which means that the ϕ should always hold.

In this work, given that we employ simulation-based approaches that return finite traces, it may happen that we cannot conclude the satisfaction or the violation of an LTL formula (that involves the until operator) from finite traces. This happens, e.g., for $a \;\mathtt{U}\; b$, when the finite system execution always satisfies a without ever satisfying b. In such a case, we conclude that the trace does not satisfy the property. We discuss ideas to improve our method in such cases in Sect. 8.

2.3 Statistical Model Checking

Statistical Model Checking (SMC) is a family of algorithms to estimate the probability that a stochastic system satisfies an LTL property ϕ [34]. The idea is to sample a set E of bounded executions of the system and to associate each execution $e \in E$ with a Bernoulli variable b_e (1 if the execution satisfies the property, 0 otherwise). Then, one can estimate the overall probability that the system satisfies ϕ as $\sum_{e \in E} \frac{b_e}{|E|}$. SMC also applies to non-stochastic systems by assuming an implicit uniform probability distribution on each state successor.

Recently, Delahaye et al. [3,15] proposed an SMC approach to verify parametric Markov chains, that is, DTMCs whose transition probabilities depend on numeric parameters function such as $Pr : S \times S \rightarrow Poly(\mathbb{X})$ where $\mathbb{X}$ is the set of parameters $p_0, ..., p_n$. Then the – parametric – function $f \in Poly(\mathbb{X})$ of a k-length path $\rho = s_0, s_1, \ldots, s_k$ in the pMC is given by $Pr_{(s_0,s_1)} \otimes \cdots \otimes \mathrm{Pr}_{(s_{k-1},s_k)}$. The probability to execute this path can be derived for any DMTCs by valuate the parameters of this function. Given a parameter valuation $\nu \in \mathbb{R}^{\mathbb{X}}$ and a parametric function $f \in Poly(\mathbb{R})$, $f(\nu)$ is the probability that the variant ν executes the path.

This approach is interesting because it samples paths in the DTMC uniformly and accumulates a parametric function that encodes the – parametric – probability of this path to be executed for any valuation (aka variants). The approach also associates every sampled execution with a reward (e.g., 1 if the execution satisfies the checked property; 0 otherwise). Then, the probability that a given parameter valuation satisfies the property is estimated as the average of all rewards weighted with the value of the associated parametric function corresponding to the parameter valuation. Delahaye et al. [15] demonstrate the soundness of their method theoretically and experimentally on three examples.

There are three fundamental differences between these parametric Markov chains and FDTMCs that impede the direct application of Delahaye et al.'s approach. First, FDTMCs include Boolean parameters, whereas parametric Markov chains include real parameters. Second, these Boolean parameters represent VIS features and determine the existence of transitions within the different variants, whereas Delahaye et al. assume that all transitions are available regardless of any particular parameter values. Third and last, VIS features are interdependent and it is necessary to filter out feature combinations that do not correspond to any existing (valid) variants. However, these two approaches can be both used to verify every variants (or valuations) of a stochastic VIS because:

$$FDTMC_{|v \in \mathbb{B}^F} = pMC_{|\nu \in \mathbb{R}^{\mathbb{X}}} \iff \forall t \in T, \varPi_t(v) = Pr_t(\nu).$$

Therefore, our work takes inspiration from the principles of Delahaye et al. [3,15] but develops a novel SMC approach to verify FDTMCs. The implementation of our algorithms relies on a dedicated data structure – based on Algebraic Decision Diagrams (ADDs) [2] – to account for the binary nature and relationships of FDTMCs parameters. Indeed, the advantage of our data structure over Delahaye's parametric approach is that ADDs can record (1) which variants can (or cannot) execute a given FDTMC path and (2) with which probability – and it can do so while keeping its structure concise as it accumulates more probability profiles. The other advantage of our approach is that we can directly embed constraints between the features within the decision diagram that can also act as constraints between transition probabilities. [8,12]. By doing so, we discard invalid combinations of features by constructions – whereas parametric approaches would invoke a solver to determine the set of valid combinations. Overall, our solution fits specifically well to the problem of verifying stochastic VISs, whereas Delahaye's method is more appropriate for classical parametric models.

3 Statistical Model Checking for Featured DTMC

We consider the problem of checking an LTL formula on a featured discrete-time Markov chain. The traditional SMC method of Younes et al. [34] can work only on a single variant. One straightforward way to address our problem is, therefore, to compute the projections of the FDTMC onto each variant $\forall v \in V, FTMC_{|v}$ and apply traditional SMC to each resulting DTMC. For example, the projection in Fig. 2 results in 6 states with two states (5 and 6) that accept the LTL formula ϕ. The traditional SMC method will sample n paths of k steps (arbitrary values). In this example, some possible paths of 3 steps are: $\rho_1 = (0, 1, 3, 2)$, $\rho_2 = (0, 2, 2, 4)$, $\rho_3 = (0, 2, 4, 3)$, $\rho_4 = (0, 1, 0, 1)$, $\rho_5 = (0, 1, 3, 5)$ and $\rho_6 = (0, 1, 3, 6)$. The two last paths ρ_5 and ρ_6 reach an accepting state. This mean that they violate the safety property (i.e., $\rho_{i \in \{5,6\}} \not\models \phi$). The probability that the system will produce these behavior is $P[\rho_5] = .5 \times .8 \times .1 = .04$ and .12 for ρ_6. More probable paths are for example ρ_1 where $P[\rho_1] = .24$ or ρ_3 (.35).

Given 6 samples P of 3 steps, lets say the one described $\rho_1, ..., \rho_6$, the probability $\mathbb{E}$ that the system violates the property is

$$\mathbb{E} = \frac{\sum_{\rho \in P, \rho \not\models \phi} P[\rho]}{\sum_{\rho \in P} P[\rho]} = \frac{.04 + .12}{.24 + .105 + .35 + .05 + .04 + .12} = 0.176$$

Although this procedure is simple, it suffers from the exponential blow-up inherent to variability [9], that is, the number of variants tends to increase exponentially in the number of features.

Instead, we propose a new SMC method that can sample executions from all variants at once (i.e., directly in the FDTMC) regardless of the probabilistic differences across the variants. We rely on the theoretical results of Delahaye et al. [3,15] and adapt their principles to FDTMC verification. Hence, our algorithm performs a uniform random walk to sample a path in the FDTMC. That is, at each step, the next transition to execute is selected uniformly regardless of the number of variants that can execute it and with which probability.

For each sample path ρ, we record three pieces of information:

- The probability profile $\mathit{\Pi}_\rho$ associated to ρ, which records the probability that each variant executes it. In our implementation, we encode a probability profile into an ADD. We also record the set of variants V_ρ that can execute ρ. We can compute this set from $\mathit{\Pi}_\rho$ a posteriori, such that $v \in V_\rho \Leftrightarrow \mathit{\Pi}_\rho(v) > 0$.
- A Boolean value b_ρ that indicates whether ρ satisfies or not the checked formula, that is, the reward of ρ according to Delahaye et al.'s terminology.
- The probability p_ρ that the uniform random walk samples ρ.

A number n of repeated applications of Algorithm 1 produce a set of tuples $\{(\rho_1, \mathit{\Pi}_1, b_1, p_1) \dots (\rho_n, \mathit{\Pi}_n, b_n, p_n)\}$. Then, leaning on Delahaye et al.'s theory [15] for parametric DTMC estimation, we can compute an estimator of the probability that any system variant satisfies the property. For a given variant v, this estimator is the average reward of the paths ρ_i sampled from the FTDMC projection onto v, weighted by the probability that v executes ρ_i, that is, $\mathbb{E}_{b_v} = \frac{1}{n_v} \times \sum_i (b_i \times \mathit{\Pi}_i(v))$ where n_v is the number of paths sampled in the projection onto v such as $\forall v \in V, n_v = \sum_{i=1}^{n} \mathit{\Pi}_i(v) > 0$. The idea is that for a large number n of samples, this expected reward converges towards the real probability that v satisfies the property. Our uniform random walk approach, however, cannot directly produce such an estimator because it samples paths uniformly, irrespective of the real probabilities that the variants can execute these paths. We, therefore, follow Delahaye et al.'s parametric approach and *normalize* the weighted average reward $\mathbb{E}_{b_v}$ with the probability with which the random walk sampled each path ρ_i. Hence, we compute the estimator as

$$\tilde{\mathit{\Pi}} = \frac{\mathbf{1}}{\mathbf{n}} \otimes (\frac{b_1}{p_1} \otimes \mathit{\Pi}_1 \oplus ... \oplus \frac{b_n}{p_n} \otimes \mathit{\Pi}_n) \tag{1}$$

where $\frac{\mathbf{1}}{\mathbf{n}}$ is the probability profile that associates $\frac{1}{n_v}$ to each variant v. Then, for sufficient numbers $\{n_v\}$, $\tilde{\mathit{\Pi}}(v)$ is the estimated probability that the variant

v satisfies ϕ. We can reduce the demonstration of this result to Delahaye et al.'s proofs for parametric DTMC [3,15], by transforming our FDTMC into a parametric DTMC (c.f. Fig. 3), such that each parameter corresponds to a feature and takes either the value 1 (the feature is enabled) or 0 (the feature is disabled). Then, in probability profile a positive literal over a feature f is simply encoded as f (e.g., $f \times 0.4$ corresponds to $\Pi(f) = 0.4$) and a negative literal over f is encoded as $(1 - f)$ (e.g., $(1 - f) \times 0.4$ corresponds to $\Pi(\neg f) = 0.2$). In the end, the only difference in our working assumptions – which has no incidence on the proof – is that the number of paths that each variant can execute can differ. This is taken into account with the profile $\frac{\mathbf{1}}{\mathbf{n}}$ and is equivalent to having paths that some variants executes with a zero probability. For the sake of conciseness, we do not replicate the proof of Delahaye et al. here.

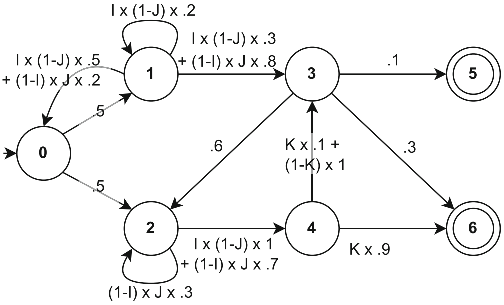


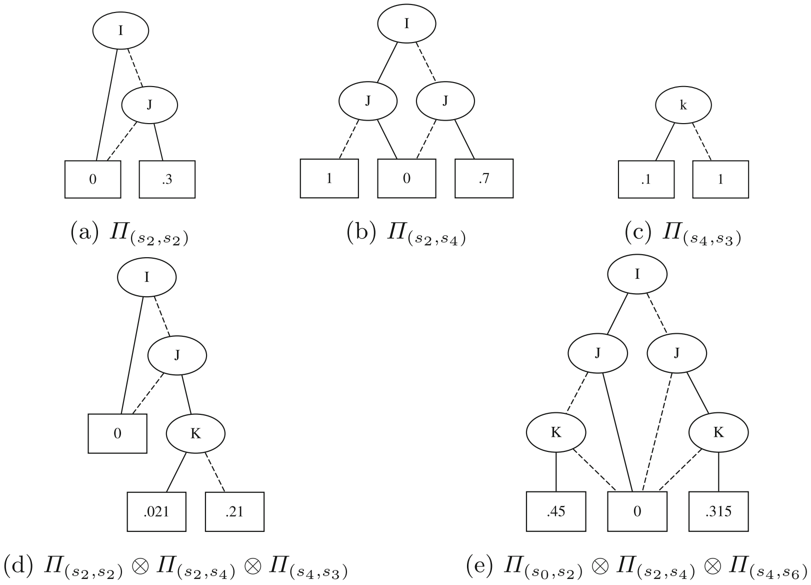
Fig. 3. Parametric DTCM resulting from the FDTMC illustrative example.

Algorithm 1 implements our random walk method for properties of the form $\Diamond a$ with $a \in AP$. The reasons we present the algorithm for these simple properties are the clarity of the presentation and because the key principle of our method is how we accumulate probability information during the random walk.

The algorithm selects an initial state of the FDTMC according to the initial state distribution ν (Line 1). Then, it enters an iterative process to select the successive states that the FDTMC goes through (Lines 7–17)). If the current state s satisfies the atomic property a then the current path ρ satisfies $\Diamond a$ – in this case, the algorithm stops and returns ρ and the associated probability profile Π_ρ that describes the probability for each variant to execute ρ (Line 19). Otherwise, the algorithm picks the next state s' from the reachable set of successors with a uniform probability (Line 12–13). In order to sample a relevant path, we consider only states s' that at least one variant can reach from s with a non-zero probability. The algorithm also updates the probability profile of ρ by multiplying it with the probability profile of the transition from s to s' (noted $\Pi_{(s,s')}$ – see Line 15). The algorithm iterates until it executes k steps or finds a state satisfying a.

The generalization to any LTL property is straightforward. It consists of executing the trace into the Büchi automaton equivalent to ϕ. Then the trace satisfies the property if and only if the execution loops over an accepting state of the automaton. Concretely, this generalization is obtained by removing Lines 9–11 and add after Line 18 the execution of the trace into the automaton. The implementation of this execution is a standard model checking procedure and is omitted here for conciseness.

The most important design decision for our algorithm is the representation we use for probability profiles and their accumulation. We propose to encode probability profiles using Algebraic Decision Diagram (ADD) data structure. ADD are a generalization of Binary Decision Diagram (BDD). BDDs are traditionally used for non-stochastic VISs and encode efficiently [9] which variants can or cannot execute the transitions. We propose ADD to also encode variants probability to execute transitions and thus to capture VISs with stochastic nature. More formally, an ADD represents $\mathbb{B}^n \rightarrow \mathbb{R}$ function. Such function can capture the probability to execute a transition for every variant. $ADD_{s_1,s_3}(I, \neg J, K) = .3$ while $ADD_{s_1,s_3}(\neg I, J, \neg K) = .8$, etc. Operations such as sum, product, modulo, etc. can be applied to ADDs. In our case, the product of two ADDs will multiply the probability values for every variants $\forall v \in V, (ADD_1 \otimes ADD_2)(v) = ADD_1(v) \otimes ADD_2(v)$. Scalar value i can also be considered as an ADD such as $\forall v \in V, ADD(v) = i$.



(a) $\Pi_{(s_2,s_2)}$

(b) $\Pi_{(s_2,s_4)}$

(c) $\Pi_{(s_4,s_3)}$

(d) $\Pi_{(s_2,s_2)} \otimes \Pi_{(s_2,s_4)} \otimes \Pi_{(s_4,s_3)}$

(e) $\Pi_{(s_0,s_2)} \otimes \Pi_{(s_2,s_4)} \otimes \Pi_{(s_4,s_6)}$

Fig. 4. Probability profiles of some transitions and paths encoded as ADDs.

Figures 4a, 4b and 4c illustrate probability profiles of some *featured* stochastic transitions as ADDs. While the (s_2, s_2) transition is only possible for variants with $\{\neg I \wedge J\}$. Such variants have .3 probability to fire this transitions (.7 to fire another one). Other variants cannot fire this transition. The (s_2, s_4) is a XOR ADD over I and J. $I \wedge \neg J$ variants can and will fire this transition as the probability is 1. Similarly, (s_4, s_3) transition is mandatory for K variants while $\neg K$ variants have .9 probability to not fire it. Figure 4d and 4e are two examples of probability profile of path (s_2, s_2, s_4, s_3) and (s_0, s_2, s_4, s_6). The first one is the product of the probability profiles of transitions illustrated in Fig. 4.

Figure 5 Illustrates our method with two sampled paths using Algorithm 1. The first path is: $\rho_1 = (s_0, s_1, s_3, s_2)$. The second is $\rho_7 = (s_0, s_2, s_4, s_6)$. As explained, Algorithm 1 returns a triplet containing the path probability profile (i.e., encoding the probability to execute the path given each variant. The Boolean value that indicates if the path violates or not the given formula $\rho_1 = 0$ while $\rho_7 = 1$, and the uniform probability to sample the path (.055 for ρ_1 and .125 for ρ_7). Then for each samples path, the average reward to violate the formula is:

$$\frac{\mathbf{1}}{\mathbf{n}} \otimes \left(\frac{0}{.055} \otimes \mathit{\Pi}_1 \oplus \frac{1}{.125} \otimes \mathit{\Pi}_7\right).$$

The profile $\mathbf{n}$ is the number of executable sampled paths for each variant. For instance, while $\{\neg I \wedge J \wedge K\}$ or $\{I \wedge \neg J \wedge K\}$ variants or can executes the two sampled paths, $\{\neg I \wedge J \wedge \neg K\}$ or $\{I \wedge \neg J \wedge \neg K\}$ can only execute ρ_1.

Note that in this example, the final (rightmost) ADD does not represent a probability (the terminal values are greater than one). This is because the number of samples is insufficient for these values to converge to the real probability values. Here, the terminal values are greater than one because the corresponding variants have a real probability to sample the paths that is greater than the probability of sampling these paths uniformly. Through additional repetitions of our algorithm, we would likely obtain other paths that the variants are less likely to execute and, ultimately, the final ADD will converge. Nevertheless, this example already illustrates the capability of our family-based approach to exploit common behaviour shared across multiple variants: by sampling only two paths in the FDTMC, we manage to get information about four different variants.

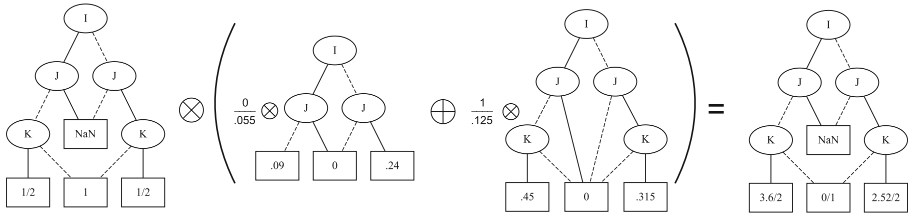


Fig. 5. Illustration of the Eq. 1, $\tilde{\mathit{\Pi}} = \frac{\mathbf{1}}{\mathbf{n}} \otimes (\frac{b_1}{p_1} \otimes \mathit{\Pi}_1 \oplus ... \oplus \frac{b_n}{p_n} \otimes \mathit{\Pi}_n)$ with two sampled paths.

```
**Input**: An FDTMC $m = (S, \nu, V, \Pi)$;
An LTL formula $\phi = \Diamond_k a$.
**Output**: a random path $\rho$ with maximum length $k$;
a binary variable $b$, equal to 1 if and only if $\rho \models \phi$;
$\Pi$, the probability profile of $\rho$;
$p_\rho$ is the probability that uniform sampling returns $\rho$
$s \leftarrow$ pick from $S$ with probability $\nu_s$;
$depth \leftarrow 0$;
$b \leftarrow \mathbf{0}$;
$\rho \leftarrow s$;
$\Pi_\rho \leftarrow \mathbf{1}$;
$p_\rho \leftarrow 1$;
while $b = \boldsymbol{0} \land depth < k$ do
    $depth \leftarrow depth + 1$;
    if $s \models a$ then
        $b \leftarrow \mathbf{1}$;
    end
    $Succ \leftarrow \{s' \in S | \exists v \in V : \Pi_{(s,s')}(v) > 0\}$;
    $s' \leftarrow$ pick from $Succ$ with probability $\frac{1}{|Succ|}$;
    $\rho \leftarrow \rho s'$;
    $\Pi_\rho \leftarrow \Pi_\rho \otimes \Pi_{(s,s')}$;
    $p_\rho \leftarrow p_\rho \times \frac{1}{|Succ|}$;
    $s \leftarrow s'$;
end
return $(\rho, b, \Pi_\rho, p_\rho)$
```

Algorithm 1: Uniform Random Walk in FDTMC

4 Evaluation

4.1 Objectives and Methodology

We conduct experiments that assess the effectiveness of our method in estimating correctly the probability of each variant to satisfy given properties. Our experiments consider a scenario where engineers have a limited simulation budget, that is, a number of SMC runs (in our case, a run is a uniform random walk). When we apply SMC to each variant, we equally share the budget between all variants. We decompose our evaluation in three research questions.

Our first research question evaluates the soundness of our approach. It aims to validate that our approach is consistent with (1) classical SMC applied to each variant's DTMC separately and (2) the parametric SMC approach of Delahaye et al. [3,15].

RQ1: Is our approach consistent with other probability estimation methods?

To demonstrate this consistency, we reuse a toy example, called "Parametric Toy", that comes from Delahaye et al.'s paper [15]. It is a simple parametric model that we transformed into an equivalent FDTMC where each variant corresponds to a parameters valuation. We consider 26 such different variants. The reason we use this example is that its small size gives us confidence that our translation has preserved the original semantics of the model. We set the simulation bound k to 15, which is sufficient to for this small model. The setting of k depends on various factors such as the ones mentioned in Sect. 4.2.

To compare the approaches, we compute the correlation between (1) our ground truth containing the probability of each variant to satisfy the given property and (2) the same probabilities estimated by the method. We computed this ground truth from a very large number of simulations run for each separate variant to precisely estimate the probabilities (10^6 simulations per variant). We use the Kendall coefficient to measure the correlation because, as an ordinal association metric, it focuses on how well each method ranks the variants according to their probability (irrespective of the actual probability values). A high Kendall correlation means that the method preserves the ranking of the variants according to their probability to satisfy the property. To complement our analysis, we also use the Pearson's correlation coefficient, which captures linear relationships between two variables (here, the ground truth versus the probability values estimated by each method). Thus, a high Pearson correlation would indicate that the method can also preserve the difference in probability between the variants.

Our second research question evaluates the benefits of factorizing the analysis over all variants at once:

RQ2: Does family-based analysis improve effectiveness?

To answer this question, we compare the effectiveness of (1) our family-based approach with (2) our uniform random walk applied to each variant separately. Since both approaches use the same sampling strategy (uniform sampling), any observable difference would show the benefit of factorizing the sampling across all variants.

We measure effectiveness as the capability of each method to estimate properly the satisfaction probability of these variants. As mentioned before, we get an aggregated view of effectiveness by computing the Kendall and Pearson correlations. That is, we measure the capability of SMC methods to rank variants properly (using Kendall) and to preserve the relative difference between the estimated probability of the variants (using Pearson).

To conduct these experiments, we use two models that are bigger than Parametric Toy. The first is the Body Sensor Network from [24] and the second is a minepump VIS [9,20]. We check both models against the property originally described in their respective papers. As for the simulation bound k, we set it to 30 for the Body Sensor Network and to 50 for the Minepump. Those values are sufficient to find violations of the properties.

Finally, we check how our uniform sampling approach compares to a product-by-product sampling that considers the transition probability values to guide the random walk (i.e., the standard way to apply SMC to probabilistic systems). The difference between this "guided" random walk and uniform random walk is that the former asymptotically produces better estimates since it directly samples from the DTMC transition function. However, this guided approach may miss rare property violations in case of insufficient sampling budget, which may affect the conclusions of SMC. Hence, our research question is:

RQ3: Is guided random walk more effective than uniform random walk?

To answer this question, we repeat our RQ2 experiments (same protocol and settings) using SMC with guided random walk applied to each variant individually.

4.2 Datasets and Parameters

To conduct our experiments, we use two models from the literature. The first is the Body Sensor Network from [24]. This system consists of a set of medical sensors that monitor a person's vital signs. Sensors and other elements exhibit variation points from which 10 variants can be derived. This system has only one property to satisfy, which is that it should never reach a failure state.

The second model is a minepump VIS [9,20] with 448 variants. The underlying FTS comprises 250,561 states[1]. We have modified this model to introduce probability over non-deterministic transitions (e.g. those that modify the level of water and the presence of methane), such that the probability mass is distributed uniformly over alternative transitions. For this model, we consider the four properties described in Table 1.

We set respectively, the simulation bound to 30 for BSN and 50 for minepump. The setting of the simulation bound k often depends on the case study. It requires knowledge about the case study but also about external factors such as system requirements, system execution context, etc. This expertise usually came from the system engineers. For example, k may depend on the property to check, such as what is the probability of having a system failure by some given time t. That is, it requires knowledge about how long the system should run before it can be considered safe.

Higher simulation bound allows finding more (but usually less probable) violations of the properties. The simulation bound may also depend on the system itself and its future usage. A more complex system will likely require a higher simulation bound because a bigger state space must be checked. If interested in rare events or if the system will run during a long time period before maintenance, the simulation bound can be increased as well. Similarly, the number of samples (or simulation runs) also depends on the case study. Basically, a complex system with a lot and/or longer paths will likely require a more significant number of samples to have a precise enough idea of the probability to satisfy the property.

[1] The state space of all variants taken individually reaches 889,124 states.

Table 1. Minepump properties that we use in our experiments.

Property number	Property formula
minepump #16	$\neg((\Box\Diamond methane) \wedge (\Box\Diamond\neg methane))$
minepump #18	$\Box(methane \Rightarrow (\Diamond stateMethanestop))$
minepump #26	$\Box((highWater \wedge \neg methane) \Rightarrow \Diamond pumpOn)$
minepump #30	$\neg\Diamond\Box(\neg pumpOn \wedge highWater)$

4.3 Implementation

We implemented our family-based SMC algorithm and the classical product-by-product SMC algorithm into ProVeLines[2] [12], a state-of-the-art model checker for VIS. The tool takes as input (a) an FDTMC – modelled in an extension of the Promela language [19] where transitions are associated with a probability distribution and can be guarded with features, (b) an LTL formula, and (c) a s sample budget of k steps. Then it runs simulations (using the available budget) and returns the probability of variants that satisfied the property. Therefore, we compare our algorithm and classical SMC on common technical ground.

To efficiently encode the probability profiles Π_ρ that our algorithm manipulates and returns, we extended the Algebraic Decision Diagrams [2] data structure. ADDs are like binary decision diagrams [5] except that leaf nodes can receive a real value. In our case, branches represent features of the VIS. Therefore, an ADD path represents a set of variants, and the value of the leaf is the probability associated to this set. Equation 1 is implemented by using two extended ADDs. ADD_p that iteratively capture the output of each path sampled by Algorithm 1 (i.e., $\frac{b}{p} \otimes \Pi_\rho$) and $ADD_\sim$ that accumulate them such $ADD_\sim = \sum_{p=0}^{n} ADD_p$. We implemented our ADD extension into the efficient CuDD library [29]. The extension allows to store multiple real values as leaves in order to optimize the implementation.

As for the parametric approach, we reuse the prototype Python implementation of Delahaye et al.[3] [15]. Unfortunately, this prototype does not support concurrent systems (i.e., multiple processes/modules) and a very limited subset of the Prism language. Consequently, it cannot verify Body Sensor Network nor Minepump case studies.

We run all our experiments on a Dell Latitude i7 16 GB 1.8 GHz. To account for random variations, we execute 10 runs of each experiment and report the average accuracy. Body Sensor Network and Minepump ground truth computations took, respectively, 6 min and 50 min approximately. The different case studies took a few seconds for Body Sensor Network and few minutes for Minepump. The memory consumption of ProVeLines was around a dozen of MB. We do not notice differences in performance between the methods implemented in ProVeLines.

[2] https://bitbucket.org/SamiLazregSuidi/provelines-stc/src/master/.

[3] https://github.com/paulinfournier/MCpMC.

5 Results

We present our experimental results for the three research questions hereafter.

5.1 RQ1: Soundness

We show in Table 2 the correlation analysis for the Parametric Toy exemplar, the only model we could model equivalently in Delahaye et al.'s tool [15] and in ours. We show the correlations for Delahaye et al.'s approach (*Parametric*), our family-based algorithm, and the product-by-product uniform random walk. For each method, we show the correlation between the probability values that the method estimated and the ground truth. As a reminder, the ground truth was computed from a very large number of simulations run for each variant to precisely estimate the probabilities (10^6 simulations per variant). Interestingly, we observe that each method achieves extremely strong correlations (above 0.89). This indicates that both the approach of Delahaye et al. [15] and our novel SMC algorithm can produce suitable estimates to compare and rank the variants of the Parametric Toy.

We can observe that the variation of the number of simulations does not drastically influence the correlation of any of the methods. This indicates that even the smallest simulation budget we considered (10^3) is enough to cover rare behaviors specific to few variants of the Toy exemplar. This also suggests that our uniform normalisation function is adequate for this system [15].

> These positive results indicate that our approach can produce suitable estimations. This allows us to have confidence in its correctness and capabilities.

This successful preliminary validation encourages us to pursue our endeavour on larger models and more complex properties, which we investigate in the next research questions.

5.2 RQ2: Benefits of Family-Based

We study the correlations achieved by our approach ("Family-based") compared to the product-by-product random walk ("PbyP: Classical"), on larger models than the parametric toy exemplar. Table 3 shows the results for Body Sensor Network (BSN) and for Minepump. It has to be noted that the two models have significantly different characteristics. The state-space of Minepump is larger than BSN and requires, therefore, longer explorations (in terms of simulation bound k) to sample relevant path prefixes. BSN, however, presents an extreme factor of complexity: all its stochastic transitions are featured, i.e., the transition probabilities change with the system features. This means that the stochastic behaviour of two variants can differ significantly and do so early during the simulation.

In the BSN case (Table 3), we observe that, for all sampling budgets, both approaches achieve very strong Pearson correlation (>0.93) and strong Kendall correlation (>0.77). This means that they are both effective in estimating the ranking of variants (wrt. their property violation probability) and even more in estimating the relative differences (in violation probability) between these variants.

In both cases, increasing the budget improves the correlation values. This indicates that, in spite of its reasonable state space size, BSN remains challenging to simulate due to the divergence in the variants' stochastic behaviour. We actually observe that, though our family-based approach strongly correlate with the ground truth, the product-based alternative is better in this case.

In the Minepump case (Table 3), we observe again that the two approaches are overall effective: they both achieve very strong Pearson correlation (>0.90) and strong Kendall correlation (>0.64) for all properties and sampling budget. There are, however, observable differences between the two techniques.

First, the family-based approach achieves better Kendall correlations than the product-by-product method (up to 0.13 difference), but worse Pearson correlation (up to 0.07). This means that the product-by-product estimates the relative difference between variants slightly better. However, it has more failures when it comes to ranking these variants. This can be explained by the fact that, in Minepump, multiple variants can have very close violation probabilities. In this case, the product-by-product method can fail to rank them due to the inherent estimation error. By contrast, the family-based approach estimates the probability of these variants at once; therefore, it applies the same estimation error to all variants, which does not impact the rankings.

Second, increasing the sampling budget increases the correlation values for the product-by-product method, but has no significant effect on the family-based approach. This indicates that our method can provide its maximal benefits even with a small sampling budget. We explain these results by the fact that the minepump variants differ more by their unique transitions than in their probabilities to execute common transitions. In such a case, the factorization capability of our family-based method is optimally used and enables the production of accurate estimation even with a low number of sampled paths.

> The main benefit of our approach is to effectively estimate the ranking of the different variants according to the probability to violate the property. Our method is more effective at ranking variants than product-by-product approaches, especially at low sampling budgets.

5.3 RQ3: Guided Sampling

Our last research question investigates whether a product-by-product sampling approach that is guided by transition probabilities brings benefits over uniform sampling. Results for this approach are again shown in Table 3 (column "PbyP: Guided").

In the BSN case, we observe that this new approach achieves very strong Pearson correlation (>0.96) – it is as effective as the product-by-product uniform sampling approach – and strong Kendall correlations (>0.73) – though, overall, less strong than the two uniform approaches. Interestingly, we observe that these Kendall correlations can significantly increase or decrease with the sampling budget, whereas one would expect the correlation to improve monotonically with the sampling budget. Because it is guided, this sampling approach inherently favours common execution paths over rare paths. In case these rare paths are violating, this biased sampling introduces random factors in the ranking of the variants.

In the Minepump case, the guided approach achieves lower Pearson and Kendall correlations compared to the two uniform approaches, though these correlations remain strong (>0.73). As we previously observed on the uniform product-by-product method, the guided approach improves its estimations with an increasing the sampling budget, though it never manages to perform better than our family-based method.

Our uniform sampling method performs better at low sampling budgets. This indicates that guided sampling methods tend to be more sensitive to path rarity, which can deteriorate the estimations.

These results demonstrate the importance of path rarity in accurately estimating all variants' violation probability. This importance, in turn, motivates the use of uniform sampling and normalization methods that we have proposed in this paper.

Table 2. Correlation between each method and the ground truth (10^6 simulations run for each variant). In each cell, left number is Pearson's correlation; right number is Kendall's. In the table, "Parametric" refers to Delahaye et al.'s approach. "Family-based" is our novel SMC algorithm. "PbyP: Classical" means uniform random walk SMC applied to each variant separately.

	Sample budget	Family-based		PbyP: Classical		Parametric	
Param. Toy	10^3	0.9655	0.9294	0.9964	0.8928	0.9570	0.9983
Param. Toy	10^4	0.9630	0.9230	0.9972	0.9733	0.9539	0.9984
Param. Toy	10^5	0.9627	0.9231	0.9998	0.9733	0.9539	0.9984

6 Threats to Validity

The first threat to validity is the models we use. We used only two VIS models from the literature that we could easily adapt to become stochastic VIS. These models do not exhibit a real-world level of complexity (i.e., hundreds of variants and hundreds of thousands of states). But even if the case studies are relatively simple compared to real-world VISs, the preliminary results of

Table 3. Correlation between each method and the ground truth (10^6 simulations run for each variant). In each cell, left number is Pearson's correlation; right number is Kendall's. In the table, "Family-based" is our novel SMC algorithm. "PbyP: Classical" means uniform random walk SMC applied to each variant separately. "PbyP: Guided" means guided SMC applied to each variant separately.

	Sample budget	Family-based		PbyP: Classical		PbyP: Guided	
BodySensorNet.	10^2	0.9386	0.7778	0.9675	0.809	0.9673	0.7333
BodySensorNet.	10^3	0.9594	0.7778	0.9984	0.8667	0.9976	0.9556
BodySensorNet.	10^4	0.9672	0.9111	0.9999	0.9111	0.9982	0.9111
BodySensorNet.	10^5	0.966	0.9556	1.0	0.9556	0.9994	0.8222
minepump#16	250	0.9586	0.778	0.9762	0.6412	0.8284	0.756
minepump#16	500	0.9617	0.7791	0.9873	0.6753	0.8274	0.7601
minepump#16	1000	0.9602	0.7697	0.9922	0.6852	0.8292	0.7606
minepump#16	2000	0.9603	0.7766	0.9953	0.7152	0.8292	0.761
minepump#18	250	0.9574	0.7651	0.9856	0.7171	0.837	0.7424
minepump#18	500	0.9576	0.7775	0.991	0.737	0.8366	0.7544
minepump#18	1000	0.958	0.7699	0.995	0.7752	0.8381	0.7518
minepump#18	2000	0.9578	0.7823	0.9969	0.781	0.837	0.7563
minepump#26	250	0.909	0.8434	0.9509	0.7841	0.8445	0.733
minepump#26	500	0.9085	0.8453	0.9677	0.7912	0.8493	0.7579
minepump#26	1000	0.909	0.8445	0.9751	0.7958	0.8526	0.7804
minepump#26	2000	0.9085	0.8469	0.9809	0.811	0.8552	0.7832
minepump#30	250	0.9027	0.8583	0.9591	0.7859	0.7799	0.7498
minepump#30	500	0.9028	0.8719	0.9745	0.8277	0.7847	0.8074
minepump#30	1000	0.9014	0.8788	0.9773	0.833	0.7856	0.8064
minepump#30	2000	0.9038	0.8765	0.9832	0.8365	0.7859	0.8323

our evaluation show that our method is a promising direction for verification of VISs. As future work, we plan to collect several real-world stochastic systems from different industries such as space, automotive and biomedical to further develop our method.

Our second threat to validity is that the effectiveness of our method may depend on the system to verify. Indeed, a basic assumption is that systems with many common behaviors across the variants will provide better results. This is what we can observe with our models. The effectiveness of our method also depends on the property to check. For example, our method provides worst Pearson but better Kendall correlations for the last two minepump properties #26 and #30. The other case studies share similar results. However, further research and investigations will be required to characterize how different systems and properties will impact our method.

Similarly, the choice of the normalization function and the simulation bound k may highly impact the results of our method. We show that the uniform normalization function outperforms the traditional guided sampling in giving accurate estimations. However, as the convergence speed can be affected by the choice of the normalization function [15], this choice may also depend on the case study. For the simulation bound, our k settings find violations for every variant with reasonable probability of happening (higher than 0.0001). Indeed, in our models, a lower k produces fewer behaviors (but more probable) that violate the property, while higher k will find more violations (but with a lower probability of happening). Nevertheless, setting precisely the simulation bound (similar to simulation run length) requires knowledge about the case study and external factors such as system requirements, system execution context, etc. This expertise usually came from the system engineers.

Another threat concerns the way we computed out ground truth that is used as real probabilities that each variant has to violate the property. We propose to compute this ground truth using a very high number of samples compared to the one used for the experiment. For example, for minepump, 10^6 samples were used to compute the ground truth. In comparison, 20^2 samples is the budget to assess a method. We also repeated the computation of ground truth multiple times to see if this high number is sufficient to avoid random variations. We observed slight probability variations from 10^{-6} to 10^{-9} (depending on the case study). These variations are too small to impact product ranking. Another way to compute the ground truth is to apply exhaustive bounded probabilistic model checking on each variant. This computation method differs from the one we propose in this paper and might not be a relevant comparison.

Finally, a construct validity comes from the metrics we use to measure effectiveness. Kendall and Pearson coefficients are established statistical methods to measure the correlation between two variables (here, the ground truth and the estimations). We reused standard libraries to compute them and are therefore confident that our computation is correct. Still, these coefficients assess to what extent the estimations preserve the real differences between variants (be it of raking or of value) and do not precisely reflect the estimation errors. Overall, these coefficients are meaningful if the goal is to compare products rather than get extremely accurate probability estimations.

7 Related Work

7.1 VIS Verification

There are numerous models proposed for VIS verification. For instance, Classen et al. proposed *Featured Transition System* [9] (FTS) formalism which is an automata-based model that relies on transitions labelled by a features expression. Consequently, this formalism determines which variants can exercise the transition. Using this information, the fact that variants have behaviour in common could be exploited, leading to significant speedup in terms of verification time.

Accordingly, Classen et al. proposed variability-aware *exhaustive* algorithms to model-check FTS.

In addition to FTS, other models have been extended to capture the behaviour of multiple variants such as modal transition system [30], product-line-CCS [18], featured Petri-nets [23]. Each formalism has a different syntax and semantics. Modal transition systems or modal I/O automata use optional "may" transitions to model variability. Similarly, product-line-CSS is a process algebra with alternative choice operators between two processes to model the behavior of a set of variants. All these approaches are reasonable solutions for VIS verification. However, most of them are isolated efforts that do not come with mature companion tools. Our work is, therefore, based on FTS. Ter Beek et al.'s [30] solution based on modal transition systems is another mature approach. However, it requires the use of a separate logic to link variants to their behaviour in the model, which we found to be less practical than the explicit variability information contained in FTS. This information makes it easy and efficient to determine the variants that can execute a given buggy behaviour [10].

There also exist VIS models that include probabilistic information, such as FDTMCs [27] and Markov decision processes [7]. These models come with dedicated generalization of exact probabilistic model checking algorithms to compute precise probability values to satisfy given properties. By contrast to all the above methods, our approach is non-exhaustive and samples paths from the model to estimate the probabilities of the stochastic VIS while reducing the verification effort. Our work, therefore, trades off the exactness of the verification results for an increased efficiency. This compromise is essential to verify VIS with large state space.

A related line of work concerns the selection of a DTMC from a family of candidate DTMCs. Ceska et al. [6] approach this problem from three angles: feasibility (does there exist a family member that satisfies the property), threshold (which family members satisfy the property within a given probability threshold, and which ones do not), and optimality (which family member optimizes the probability to satisfy the property). They propose a solution to answer these three questions based on an abstraction-refinement scheme over Markov decision processes. Our objectives differ in that we aim to estimate the violation probability of each family member, a more precise information that is not necessary (but is sufficient) to answer the above questions. The exploration of using SMC and combine it with Ceska et al.'s abstraction approach is an interesting direction for future work.

7.2 SMC for VIS

Recent work has applied SMC in the context of VIS. Vandin et al. [31,32], proposed an algebraic language to formally model behaviour with dynamic variability (i.e. where the system can adapt its configuration during its execution). Vandin et al. also proposed a product-based SMC approach to check properties on the reconfigurable system. Contrary to this work, our approach assumes static variability (the variants are distinct systems) and relies on family-based algorithms to reason more efficiently on the whole set of variants.

Dubslaff et al. [17] and Nunes et al. [27] have studied VIS subject to stochastic behaviour and proposed exhaustive model-checking algorithms to check the probabilistic properties of such systems. These algorithms suffer from scalability limitations because of their exhaustive nature and the inherent computational cost of stochastic model checking approaches. To overcome this scalability issue, Delahaye et al. applied SMC to stochastic parametric systems [16]. Their approach opens the possibility to verify stochastic VIS, where each variant is a valuation of the parameters. More precisely, Delahaye et al. target the verification of quantitative reachability properties. By contrast, we support non-quantitative but more general properties (expressed in a fragment of LTL).

In a series of recent works [31], ter Beek et al. proposed a simulation-based approach for software product lines with stochastic behaviours. The approach relies on an algebra to describe sets of variants and on SMC [21,22,34] to compute the probability of each variant to satisfy a given bounded LTL property.

In this paper, we reconciled the approaches of [31] and of [15] by proposing a family-based extension of SMC for FDTMC. We sketched the theory and proposed an implementation in the ProVeLines model checker [12]. We then show that our approach is more effective than the traditional, guided, product-by-product SMC method. It is, furthermore, sample-efficient as its factorization capability enables the production of suitable estimations with a low sample budget.

8 Conclusion

There are two majors difficulties with the verification of VISs. The first is to find a compact representation for a set of variants that share a common basis of behaviours, but also differ by their unique behaviours. The second is to exploit this representation to evaluate each variant efficiently.

In this paper, we consider VISs whose behaviours depend on stochastic information. As seen in [27], such systems can be represented with FDTMC. That is to say with transition systems whose transitions are extended with probability profiles. Such profiles list the set of variants that are following the transitions as well as the probability to take such a transition for a given variant.

Interestingly, we got some promising results that our family-based approach produces consistent results and could precisely estimate the rank of the different variants, especially at very low simulation budgets. Product-based approaches seem to suffer from a fundamental limitation. The amount of estimations errors sums up between the different variants reducing thus ranking capabilities. Consequently, they may require more significant simulation budgets to outperform our method.

Over the last years, verifying FDTMC has been the subject of intense studies (see e.g., [17,27]). Some of the verification techniques that have been proposed are family-based; that is, exploiting the compact structure of FDTMC to avoid redundant work. Other approaches enumerate and perform verification on each variant represented by the FDTMC. All those studies rely on extensions of probabilistic model checking algorithms. While such algorithms are precise, they eventually suffer from the state-space explosion problem.

This paper is the beginning of a new thread of results on applying SMC to VISs. There are various directions for future research. The first direction is to consider variants with both stochastic and non-deterministic aspects. This could be done by combining the result of the present paper with the smart sampling approach for non-deterministic behaviours proposed in [14]. Another extension concerns the properties that we can verify. The present paper is restricted to bounded executions. The problem is that verifying full LTL over infinite executions is incompatible with a simulation-based approach. Indeed, the main hypothesis of such an approach is that the property can be decided on each simulation after a finite number of steps. This is a contradiction with the liveness fragment of LTL that requires monitoring unbounded executions. Several authors have proposed solutions to this problem. These solutions either require to have computed to the full state space of the model, or they drastically increase the number of simulations [13,33]. We plan to investigate a novel approach based on three-valued LTL. The idea would be to use the work in [4] that offers a finite-word automata-based representation to monitor LTL properties. Given a finite execution, the approach can either decide if it satisfies the property by comparing the outcomes of two finite automata, or return an undefined value in case the comparison is inconclusive. Such a three-valued approach cannot be handled by classical Monte Carlo algorithms, but promising extensions exist [1] and can inspire our work.

References

1. Arora, S., Legay, A., Richmond, T., Traonouez, L.-M.: Statistical Model Checking of Incomplete Stochastic Systems. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11245, pp. 354–371. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03421-4_23
2. Bahar, R.I., Frohm, E.A., Gaona, C.M., Hachtel, G.D., Macii, E., Pardo, A., Somenzi, F.: Algebric decision diagrams and their applications. Formal Methods Syst. Des. **10**(2), 171–206 (1997). https://doi.org/10.1023/A:1008699807402
3. Bao, R., Attiogbe, C., Delahaye, B., Fournier, P., Lime, D.: Parametric Statistical Model Checking of UAV Flight Plan. In: Pérez, J.A., Yoshida, N. (eds.) FORTE 2019. LNCS, vol. 11535, pp. 57–74. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21759-4_4
4. Bauer, A., Leucker, M., Schallhart, C.: Monitoring of Real-Time Properties. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 260–272. Springer, Heidelberg (2006). https://doi.org/10.1007/11944836_25
5. Bryant, R.E.: Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Comput. Surv. **24**(3), 293–318 (1992)
6. Češka, M., Jansen, N., Junges, S., Katoen, J.-P.: Shepherding Hordes of Markov Chains. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11428, pp. 172–190. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17465-1_10

7. Chrszon, P., Dubslaff, C., Klüppelholz, S., Baier, C.: ProFeat: feature-oriented engineering for family-based probabilistic model checking. Formal Aspects of Computing **30**(1), 45–75 (2017). https://doi.org/10.1007/s00165-017-0432-4
8. Classen, A., Cordy, M., Heymans, P., Schobbens, P.Y., Legay, A.: Snip: an efficient model checker for software product lines. Technical report, University of Namur (FUNDP) (2011)
9. Classen, A., Cordy, M., Schobbens, P.Y., Heymans, P., Legay, A., Raskin, J.F.: Featured transition systems: foundations for verifying variability-intensive systems and their application to LTL model checking. Trans. Softw. Eng. **39**, 1069–1089 (2013)
10. Classen, A., Heymans, P., Schobbens, P.Y., Legay, A., Raskin, J.F.: Model checking lots of systems: efficient verification of temporal properties in software product lines. In: ICSE'10, pp. 335–344. ACM (2010)
11. Clements, P.C., Northrop, L.: Software Product Lines: Practices and Patterns. SEI Series in Software Engineering, Addison-Wesley (2001)
12. Cordy, M., Schobbens, P.-Y., Heymans, P., Legay, A.: Provelines: a product-line of verifiers for software product lines. In: SPLC'13, pp. 141–146. ACM (2013)
13. Daca, P., Henzinger, T.A., Kretínský, J., Petrov, T.: Faster statistical model checking for unbounded temporal properties. ACM Trans. Comput. Log. 18(2), 12:1–12:25 (2017)
14. D'Argenio, P.R., Legay, A., Sedwards, S., Traonouez, L.: Smart sampling for lightweight verification of markov decision processes. CoRR, abs/1409.2116 (2014)
15. Delahaye, B., Fournier, P., Lime, D.: Statistical model checking for parameterized models. working paper or preprint (2019)
16. Delahaye, B., Fournier, P., Lime, D.: Statistical model checking for parameterized models (2019)
17. Dubslaff, C., Klüppelholz, S., Baier, C.: Probabilistic model checking for energy analysis in software product lines. In: Binder, W., Ernst, E., Peternier, A., Hirschfeld, R., (eds.) 13th International Conference on Modularity, MODULARITY '14, Lugano, Switzerland, 22–26 April, 2014, pp. 169–180. ACM (2014)
18. Gruler, A., Leucker, M., Scheidemann, K.: Modeling and Model Checking Software Product Lines. In: Barthe, G., de Boer, F.S. (eds.) FMOODS 2008. LNCS, vol. 5051, pp. 113–131. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68863-1_8
19. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley (2004)
20. Kramer, J., Magee, J., Sloman, M., Lister, A.: Conic: an integrated approach to distributed computer control systems. Comput. Digit. Tech. IEE Proc. E **130**(1), 1–10 (1983)
21. Legay, A., Delahaye, B., Bensalem, S.: Statistical Model Checking: An Overview. In: Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G., Roşu, G., Sokolsky, O., Tillmann, N. (eds.) RV 2010. LNCS, vol. 6418, pp. 122–135. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16612-9_11
22. Legay, A., Lukina, A., Traonouez, L.M., Yang, J., Smolka, S.A., Grosu, R.: Statistical Model Checking. In: Steffen, B., Woeginger, G. (eds.) Computing and Software Science. LNCS, vol. 10000, pp. 478–504. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91908-9_23
23. Muschevici, R., Clarke, D., Proenca, J.: Feature petri nets. In: Proceedings of the 14th International Software Product Line Conference (SPLC 2010), vol. 2. Lancaster University, Lancaster, United Kingdom (2010)

24. Nunes, V., Fernandes, P., Alves, V., Rodrigues, G.: Variability management of reliability models in software product lines: an expressiveness and scalability analysis. In: SBCARS '12, pp, 51–60 (2012)
25. Pnueli, A.: The temporal logic of programs. In: FOCS'77, pp. 46–57 (1977)
26. Raatikainen, M., Soininen, T., Männistö, T., Mattila, A.: A Case Study of Two Configurable Software Product Families. In: van der Linden, F.J. (ed.) PFE 2003. LNCS, vol. 3014, pp. 403–421. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24667-1_30
27. Rodrigues, G.N., et al.: Modeling and verification for probabilistic properties in software product lines. In: HASE 2015, Daytona Beach, FL, USA, 8–10 January, 2015, pp. 173–180 (2015)
28. Sabin, D., Weigel, R.: Product configuration frameworks-a survey. IEEE Intell. Syst. Appl. **13**(4), 42–49 (1998)
29. Somenzi, F.: Cudd: Cu Decision Diagram Package-Release 2.4. 0. University of Colorado at Boulder (2012)
30. ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F.: Modelling and analysing variability in product families: model checking of modal transition systems with variability constraints. J. Logical Algebraic Methods Program. **85**(2), 287–315 (2016)
31. ter Beek, M.H., Legay, A., Lluch-Lafuente, A., Vandin, A.: A framework for quantitative modeling and analysis of highly (re)configurable systems. IEEE Trans. Softw. Eng. **46**(3), 321–345 (2020)
32. Vandin, A., ter Beek, M.H., Legay, A., Lluch Lafuente, A.: QFLan: A Tool for the Quantitative Analysis of Highly Reconfigurable Systems. In: Havelund, K., Peleska, J., Roscoe, B., de Vink, E. (eds.) FM 2018. LNCS, vol. 10951, pp. 329–337. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95582-7_19
33. Younes, H.L.S., Clarke, E.M., Zuliani, P.: Statistical Verification of Probabilistic Properties with Unbounded Until. In: Davies, J., Silva, L., Simao, A. (eds.) SBMF 2010. LNCS, vol. 6527, pp. 144–160. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19829-8_10
34. Younes, H.L.S., Simmons, R.G.: Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 223–235. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45657-0_17