Symbolic Model Checking for Factored Probabilistic Models

David Deininger, Rayna Dimitrova, and Rupak Majumdar

MPI-SWS, Kaiserslautern and Saarbrücken, Germany
david.deininger,rayna,rupak@mpi-sws.org

Abstract. The long line of research in probabilistic model checking has resulted in efficient symbolic verification engines. Nevertheless, scalability is still a key concern. In this paper we ask two questions. First, can we lift, to the probabilistic world, successful hardware verification techniques that exploit local variable dependencies in the analyzed model? And second, will those techniques lead to significant performance improvement on models with such structure, such as dynamic Bayesian networks?
To the first question we give a positive answer by proposing a probabilistic model checking approach based on factored symbolic representation of the transition probability matrix of the analyzed model. Our experimental evaluation on several benchmarks designed to favour this approach answers the second question negatively. Intuitively, the reason is that the effect of techniques for reducing the size of BDD-based symbolic representations do not carry over to quantitative symbolic data structures. More precisely, the size of MTBDDs depends not only on the number of variables but also on the number of different terminals they have (which influences sharing), and which is not reduced by these techniques.

1 Introduction

Probabilistic model checking is a formal technique for analyzing finite-state models of systems that exhibit randomized behaviour against (quantitative) temporal specifications. Model checking tools, such as PRISM [13], have been successfully applied to a variety of systems, such as randomized distributed protocols, biological processes, and randomized algorithms for leader election.

State-of-the-art probabilistic model checkers such as PRISM implement symbolic model checking algorithms on top of data structures such as BDDs and MTBDDs [16]. It is well known that these data structures allow efficient sharing of state within the model checker and offer significant benefits in time and space requirements for model checking large probabilistic systems.

However, the scalability of automatic probabilistic verification remains to be a concern. A natural question is whether the structure of the probabilistic model can be exploited to provide further optimizations in state storage during model checking. For example, consider probabilistic models that exhibit local dependencies, where variables depend on a small number of "neighboring" variables.

Such local structure is common and natural in distributed algorithms and networks, in which a process communicates only with its immediate neighbours. In this setting the model can be described in a factored representation such as a Bayesian network that captures these local dependencies. Current approaches to probabilistic model checking do not benefit from the structure of the analyzed models, as this is typically lost during the translation into the verifier's internal representation as a monolithic BDD or MTBDD. Exploiting structure is identified as one of the rules of thumb in symbolic probabilistic verification [12], but most implementations only consider simple variable ordering heuristics.

In hardware model checking, one way to exploit structure is to retain the transition relation of a circuit in *partitioned* fashion [6]. Instead of computing a monolithic transition relation as a conjunction of BDDs representing modules executing in parallel, partitioned representations maintain a list of BDDs for each module. During successor computation, partitioned BDDs are manipulated one at a time using early quantification, which keeps the size of intermediate BDDs small. Partitioned approaches have been used with great success to reduce state space explosion in symbolic model checking, often by orders of magnitude [7]. It is thus natural to ask if these techniques can be successfully extended and applied to improve the efficiency of the verification of factored probabilistic models.

This is the question which we study in this paper. We have implemented a model checker for PCTL for factored probabilistic models. It accepts factored probabilistic models, in the form of dynamic Bayesian networks. These models admit a natural straightforward factored symbolic representation of their transition matrices. Our model checker uses a partitioned representation of the transition matrices as sets of MTBDDs. We extend matrix-vector multiplication based on MTBDDs to use partitioned representations of the transition probability matrix. Furthermore, we show that this procedure can be seamlessly integrated in the power method for iteratively solving systems of linear equations, which lies at the core of quantitative PCTL model checking [3, 2].

We experimentally compare the performance of PCTL model checking using partitioned versus monolithic representations on a set of scalable benchmarks that exhibit local structure. We compare our implementation against an equivalent implementation that uses a global, non-partitioned transition relation (to ensure we only capture the effect of monolithic vs. partitioned representations and do not confound our results with orthogonal heuristics). We also compare against the PRISM model checker to ensure our global representation-based implementation is comparable to the state-of-the-art.

Unfortunately, our results in the quantitative setting are negative. While *qualitative* PCTL model checking inherits the benefits of partitioned non-probabilistic model checking, we show that even on factored models, *quantitative* model checking does not significantly benefit from partitioned representations. On all but the simplest examples and properties, computing the matrix vector product on the factored representation using early variable elimination (the quantitative analogue of early quantification) does not help: while the number of variables in the MTBDD does decrease, the intermediate products have a large

number of constant terms as terminal nodes. This decreases the amount of sharing, consequently not reducing the size of the MTBDD. Overall, for quantitative specifications, partitioned representations and early variable elimination does not significantly improve run times or memory requirements over global representations. (Although, some improvement is seen on the simplest examples.)

Our negative observations carry over to different structures of the dependency graph: linear, tree, and grid topologies. The tree and grid topologies specifically were chosen to be difficult for the classical methods as there is no natural variable ordering facilitating MTBDD reasoning. With the exception of particularly easy properties that only refer to a small part of the model, these examples turned out to be hard even for the approach using the partitioned transition relation.

While our experimental results are negative, we consider them an important contribution to the research landscape in probabilistic verification. Partitioning is an intuitive heuristic, and works well in non-probabilistic settings. Our objective was to evaluate if it can be easily and naturally applied to improve the performance of probabilistic reasoning. It was surprising to us that it does not improve quantitative model checking, but to the best of our knowledge, no prior experimental comparison pointed this out. We hope that our results, showing which avenues have turned out unsuccessful, will be valuable to others aiming to improve the efficiency of probabilistic verification.

Related work. Several lines of work have investigated connections between model checking of temporal properties and inference in dynamic Bayesian networks. In [14] model checking techniques are used to perform inference in dynamic Bayesian networks for queries specified in probabilistic CTL. There, a dynamic Bayesian network is converted to probabilistic reactive modules, which are in turn encoded as an MTBDD by the PRISM model checker. Their approach does not modify the internal data structures and algorithms of the probabilistic model checker to make use of the model's structure. In [15], inference techniques are used to perform approximate model checking of dynamic Bayesian networks against finite-horizon probabilistic linear temporal properties.

2 Probabilistic Model Checking

2.1 Probabilistic Models and Temporal Logics

A *discrete-time Markov chain (DTMC)* is a tuple $M = (S, P, AP, L)$, where S is a finite set of states, $P\colon S \times S \to [0,1]$ is a transition probability function, such that $\sum_{s' \in S} P(s,s') = 1$ for every state $s \in S$, AP is a finite set of atomic propositions, and $L\colon S \to 2^{AP}$ is a labelling function mapping each state to the set of propositions that hold true in it. The transition probability function P can be interpreted as a $|S| \times |S|$ real matrix, where $|S|$ is the number of states.

A *path* in M is a finite or infinite sequence $s_0, s_1, \ldots$ of states in S such that for each i it holds that $P(s_i, s_{i+1}) > 0$. Given a state $s \in S$, we denote with $\mathsf{Paths}(M, s)$ the set of paths in M originating in the state s.

We now recall the syntax of Probabilistic Computation Tree Logic (PCTL). We fix a set AP of *atomic propositions*. The set of PCTL formulas over AP consists of two types of formulas: *state formulas* and *path formulas*. State formulas are formed according to the grammar $\Phi ::= \mathsf{tt} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Psi \mid \mathbb{P}_J(\varphi)$, where $a \in AP$, Φ_1, Φ_2 and Ψ are state formulas, $J \subseteq [0,1]$ is a real interval, and φ is a path formula. Path formulas are defined by the grammar $\varphi ::= \bigcirc\Phi \mid \Phi_1 \,\mathcal{U}\, \Phi_2 \mid \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2$, where Φ, Φ_1 and Φ_2 are state formulas, and $k \in \mathbb{N}$. As usual, we define the derived operators $\Diamond\varphi = \mathsf{tt}\,\mathcal{U}\,\varphi$ and $\Box\varphi = \neg\Diamond\neg\varphi$. The *qualitative* fragment of PCTL restricts the interval J in the probability operator $\mathbb{P}_J$ to the cases $\mathbb{P}_{=1} = \mathbb{P}_{[1,1]}$ and $\mathbb{P}_{=0} = \mathbb{P}_{[0,0]}$.

The semantics of PCTL with respect to Markov chains is defined as follows. Let $M = (S, P, AP, L)$ be a DTMC. Then, PCTL state formulas are interpreted over states of M, while path formulas are interpreted over paths. The satisfaction relations $\models$ are defined as usual for assertions, Boolean and temporal operators [3]. Formulas containing the probability operator $\mathbb{P}$ are interpreted using a probability measure over sets of paths. More specifically, the satisfaction of $\mathbb{P}_J(\varphi)$ in a state s is determined by the probability measure of the set of paths $\Pi_\varphi = \{\pi \in \mathsf{Paths}(M,s) \mid M, \pi \models \varphi\}$, for which it is known that it is measurable. More precisely, with each DTMC M and state s in M we can associate a probability measure Pr_s^M such that for every path formula φ the set of paths Π_φ is measurable [3]. Then, we define $M, s \models \mathbb{P}_J(\varphi)$ iff $Pr_s^M(\Pi_\varphi) \in J$.

It is well known that for the satisfaction of qualitative PCTL formulas in a finite-state DTMC $M = (S, P, AP, L)$ the precise values of the probabilities assigned by P do not play a role. We thus define the transition relation function $T : S \times S \to \{0,1\}$ such that for $s, s' \in S$ we have $T(s,s') = 1$ iff $P(s,s') > 0$. This defines the graph $G_M = (S, E)$ corresponding to M, with vertices the states of M, and set of edges $E \subseteq S \times S$ such that $(s,s') \in E$ iff $T(s,s') = 1$.

2.2 Probabilistic Model Checking

Given a DTMC $M = (S, P, AP, L)$ and a PCTL state formula Φ, the *model checking problem* asks to determine whether $M, s \models \Phi$ holds for every $s \in S$.

The model checking problem for PCTL can be solved by computing the set $\mathsf{Sat}_M(\Phi) = \{s \in S \mid M, s \models \Phi\}$ of states in M that satisfy Φ, and then checking if $\mathsf{Sat}_M(\Phi) = S$. The set $\mathsf{Sat}_M(\Phi)$ can be computed recursively in a bottom-up manner, following the syntax tree of the formula Φ. The key step is computing the set $\mathsf{Sat}_M(\Phi)$ for a formula of the form $\Phi = \mathbb{P}_J(\varphi)$, where φ is a path formula for which we have already computed $\mathsf{Sat}_M(\Psi)$ for every state subformula Ψ.

If $\Phi = \mathbb{P}_J(\bigcirc\Psi)$, we check if $(\sum_{s' \in \mathsf{Sat}_M(\Psi)} P(s,s')) \in J$. The probabilities can be computed by multiplying the probability matrix P with the characteristic vector of $\mathsf{Sat}_M(\Psi)$, i.e. a vector $(b_{s'})_{s' \in S}$ with $b_{s'} = 1$ iff $s' \in \mathsf{Sat}_M(\Psi)$.

For an until formula $\varphi = \Phi_1 \,\mathcal{U}\, \Phi_2$ or $\varphi = \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2$ we first compute sets $\widehat{S}_{=1} \subseteq \{s \in S \mid Pr(M, s \models \varphi) = 1\}$ and $\widehat{S}_{=0} \subseteq \{s \in S \mid Pr(M, s \models \varphi) = 0\}$ such that in the states in $\widehat{S}_{=1}$ the formula φ is satisfied with probability 1 and in the states in $\widehat{S}_{=0}$ it holds with probability 0. Furthermore we require that $\mathsf{Sat}_M(\Phi_2) \subseteq \widehat{S}_{=1}$ and that $S \setminus (\mathsf{Sat}_M(\Phi_1) \cup \mathsf{Sat}_M(\Phi_2)) \subseteq \widehat{S}_{=0}$. The remaining

states $S_? = S \setminus (\widehat{S}_{=1} \cup \widehat{S}_{=0})$ are the ones for which the probability has to still be computed. To this end, we define the matrix $A = (P(s,s'))_{s,s' \in S_?}$, which restricts P to states in $S_?$, and the vector $(b_s)_{s \in S_?}$, with $b_s = P(s, \widehat{S}_{=1})$.

The vector $(Pr(s \models \Phi_1 \,\mathcal{U}\, \Phi_2))_{s \in S_?}$ is the least fixed point of the operator $\Upsilon \colon [0,1]^{S_?} \to [0,1]^{S_?}$, with $\Upsilon(c) = A \cdot c + b$. This formulation can be rewritten into a system of linear equations $(I - A) \cdot c = b$, where I is the identity matrix of dimension $|S_?| \times |S_?|$. Choosing $\widehat{S}_{=0}$ to be exactly the set $\{s \in S \mid Pr(M, s \models \varphi) = 0\}$ guarantees that this system of equations has a unique solution [3].

For bounded until formulas $\varphi = \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2$ we have to take $\widehat{S}_{=1} = \mathsf{Sat}_M(\Phi_2)$, that is, the set of states that reach Φ_2 in zero steps, and can compute the vector of probabilities $(Pr(s \models \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2))_{s \in S_?}$ as the vector $c^{(n)}$, where $c^{(0)} = (0)_{s \in S_?}$ and $c^{(i+1)} = \Upsilon(c^{(i)})$ for $i \geq 0$. Finally, $\mathsf{Sat}_M(\Phi) = \{s \in S \mid Pr(M, s \models \varphi) \in J\}$.

Thus, computing the set $\mathsf{Sat}_M(\Phi)$ for a quantitative formula $\Phi = \mathbb{P}_J(\varphi)$ is reduced to computing the sets $\mathsf{Sat}_M(\mathbb{P}_{=1}(\varphi)) = \{s \in S \mid Pr(M, s \models \varphi) = 1\}$ and $\mathsf{Sat}_M(\mathbb{P}_{=0}(\varphi))) = \{s \in S \mid Pr(M, s \models \varphi) = 0\}$ for the respective qualitative formulas and then solving a system of linear equations.

The sets $\mathsf{Sat}_M(\mathbb{P}_{=1}(\varphi))$ and $\mathsf{Sat}_M(\mathbb{P}_{=0}(\varphi))$ do not depend on the exact values in P and can be computed based on the graph $G_M = (S, E)$ associated with M.

The set $\mathsf{Sat}_M(\mathbb{P}_{=0}(\Phi_1 \,\mathcal{U}\, \Phi_2))$ can be computed by first computing the set of states $\mathsf{Sat}_M(\mathbb{P}_{>0}(\Phi_1 \,\mathcal{U}\, \Phi_2))$ backward reachable from $\mathsf{Sat}_M(\Phi_2)$ by visiting only states in $\mathsf{Sat}_M(\Phi_1)$, and then taking $\mathsf{Sat}_M(\mathbb{P}_{=0}(\Phi_1 \,\mathcal{U}\, \Phi_2)) = S \setminus \mathsf{Sat}_M(\mathbb{P}_{>0}(\Phi_1 \,\mathcal{U}\, \Phi_2))$. The bounded until case is analogous.

The set $\mathsf{Sat}_M(\mathbb{P}_{=1}(\Phi_1 \,\mathcal{U}\, \Phi_2))$ can also be computed by backward reachability in a graph modified as follows. Let $G'_M = (S, E')$ be obtained from G_M by making all states in the set $D = \mathsf{Sat}_M(\Phi_2) \cup (S \setminus (\mathsf{Sat}_M(\Phi_1) \cup \mathsf{Sat}_M(\Phi_2)))$ absorbing. That is, $(s, s') \in E'$ iff $s \notin D$ and $(s, s') \in E$, or $s \in D$ and $s = s'$. Then, as shown in [3], it holds that $\mathsf{Sat}_M(\mathbb{P}_{=1}(\Phi_1 \,\mathcal{U}\, \Phi_2)) = S \setminus \mathsf{Pre}^*_{G'_M}(S \setminus (\mathsf{Pre}^*_{G'_M}(\mathsf{Sat}_M(\Phi_2))))$, where, $\mathsf{Pre}^*_{G'_M}(U)$ are the states backward reachable from the set U in G'_M.

2.3 Symbolic Model Checking

Let $M = (S, P, AP, L)$ be a DTMC and suppose that X is a set of Boolean variables such that $S = \{0,1\}^X$, i.e., X is a Boolean encoding of S, and the set AP consists of atomic propositions, one for each variable in X. Let $n = |X|$, and as usual, let $X' = \{x' \mid x \in X\}$ be the set of "next state" variables for X.

The transition probability function P can be encoded as a real valued function of Boolean vectors $\rho : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{R}$, and the transition relation function T can be described by a function $\delta : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$. Similarly, sets of states and probability vectors are represented as functions from $\mathbb{B}^n$ to $\mathbb{B}$ and $\mathbb{R}$, respectively.

In symbolic verification, Boolean functions are often succinctly represented as reduced ordered binary decision diagrams (BDDs) [5]. Given a fixed total ordering of the variables in $X \cup X'$, BDDs represent the Boolean functions on $\mathbb{B}^n \times \mathbb{B}^n$ in a one-to-one manner. There exist efficient methods for computing existential abstraction, application of Boolean operators and variable renaming using BDDs. The size of the BDD representing a given function is heavily

influenced by the ordering of the variables, and it is well known that finding an optimal ordering is a hard problem. A commonly used heuristic, which performs quite well in practice, is to interleave non-primed and primed variables $x < x' < y < y' < \ldots$.

In quantitative verification, a generalization of BDDs, called multi-terminal BDDs (MTBDDs) [8] are used to succinctly represent real-valued functions. The matrix and vector arithmetic operations used in PCTL model checking can be efficiently performed on their MTBDD-based representation [10, 1].

Given a fixed variable ordering, the size of the BDD representation of a Boolean function is influenced by the number of variables on which this function actually depends. The same holds for MTBDDs, where, in addition, the number of values in the co-domain of the functions has an impact on the size of the corresponding MTBDD. In the following subsection we describe a class of probabilistic models, whose structure allows for a factored symbolic representation of its transition relation and transition probability functions. Such a factored representation is a collection of BDDs, or respectively MTBDDs, that capture local dependencies between the variables describing the model, and are, often significantly smaller than those describing the transitions between global states.

2.4 Dynamic Bayesian Networks

Intuitively, a *Bayesian network* is a graph-like representation of dependence conditions on a set of random variables, coupled with some representation of the distributions associated with these random variables. More formally, a Bayesian network over a set of variables V is tuple $B = (G, \Theta)$, where $G = (V, D)$ is a directed acyclic graph with vertices the variables in V and set of edges $D \subseteq V \times V$ describing the dependencies between these variables, and Θ is a set of conditional probability distributions (CPDs), one for each variable in V, as we now explain.

For a set of variables $Y \subseteq V$, let $Val(Y)$ be the set of valuations of the variables Y, that is, the functions that map each variable $y \in Y$ to a value in its domain $Val(y)$. With $Pa_B(v) = \{u \in V \mid (u, v) \in D\}$ we denote the set of parent nodes of v in G. These are the variables on whose value the probability distribution of v directly depends. More precisely, for each variable $v \in V$ the set Θ contains a CPD $\Theta_{v|Pa_B(v)} = Pr(v \mid Pa_B(v))$. When $Val(V)$ is finite, the CPD of each variable v is usually represented by a *conditional probability table* that maps every valuation in $Val(Pa_B(v))$ to a probability distribution over $Val(v)$.

Dynamic Bayesian networks (DBN) describe systems evolving over time. A DBN over a set of variables V is a two-slice Bayesian network $B = (G, \Theta)$ over $V \cup V'$, where $Pa_B(v) = \emptyset$ for each $v \in V$. That is, the CPDs of the variables V in B depend on none of the other variables, while the CPDs of the variables in V' can depend on variables in both V and V'. More precisely, since the dependency graph G is acyclic, the CPD of a next-state variable v' can depend on the current values of V as well as on the next-state values of variables different from v.

A DBN $B = (G, \Theta)$ over a set of variables V can be seen as a factored representation of a Markov chain. The DTMC $M_B = (S, P, AP, L)$ induced by B has set of states $S = Val(V)$. The transition probability function $P(s, s') =$

$Pr_B(X' = s' \mid X = s)$ is defined according to the probability distribution described by B. We choose $AP = S$ and define $L(s) = \{s\}$.

The *model checking problem for dynamic Bayesian networks* asks, given a DBN $B = (G, \Theta)$, whose induced DTMC is $M_B = (S, P, AP, L)$, and a PCTL state formula Φ over AP, to determine whether $M_B, s \models \Phi$ for every $s \in S$.

3 Model Checking Qualitative PCTL

3.1 Factored BDD Representation

Let $B = (G, \Theta)$ be a DBN over a set of finite-state variables V. Suppose w.l.o.g. that for each $v \in V$ it holds that $|Val(v)|$ is a power of 2. Then, with each $v \in V$ we associate a set of Boolean variables X_v such that X_v is a Boolean encoding of $Val(v)$. With X'_v we denote the set of next-state Boolean variables.

Let $X = \bigcup_{v \in V} X_v$ and $X' = \bigcup_{v \in V} X'_v$. Then, $Val(X)$ are the states of the DTMC $M_B = (S, P, AP, L)$ induced by B, and the transition relation function of M_B can be represented by a BDD $\delta(X, X')$ over the variables $X \cup X'$.

Since each $v' \in V'$ depends directly only on $Pa_B(v')$, the variables in X'_v depend directly only on $\widehat{X}_{v'} = (\bigcup_{u \in Pa_B(v') \cap V} X_u) \cup (\bigcup_{u \in Pa_B(v') \cap V'} X'_u)$. We represent each $\Theta_{v'|Pa_B(v')}$ by a BDD $\delta_v(X, X')$, whose support is $X'_v \cup \widehat{X}'_v$.

If $Pa_B(v') = \{u_1, \ldots, u_k\}$, we use $(\mathbf{v}, \mathbf{u}_1, \ldots, \mathbf{u}_k, p) \in \Theta_{v'|Pa_B(v')}$ to denote the elements of the conditional probability table $\Theta_{v'|Pa_B(v')}$ for v', i.e., the fact that that $Pr_B(v' = \mathbf{v} \mid u_1 = \mathbf{u}_1, u_2 = \mathbf{u}_2, \ldots, u_k = \mathbf{u}_k) = p$.

For each $v \in V$ and $\mathbf{v} \in Val(v)$, we denote with $\beta_{v,\mathbf{v}}(X)$ the BDD for the Boolean formula over X equivalent to the atomic predicate $v = \mathbf{v}$. Similarly for $v' \in V'$ we have $\beta_{v',\mathbf{v}}(X')$. Now, for each $v \in V$, we define the BDD $\delta_v(X, X') = \bigvee_{\mathbf{v} \in Val(v')} \left(\beta_{v',\mathbf{v}}(X') \wedge \bigvee_{\substack{(\mathbf{v}, \mathbf{u}_1, \ldots, \mathbf{u}_k, p) \in \Theta_{v|Pa_B(v)} \\ p > 0}} (\beta_{u_1,\mathbf{u}_1}(X) \wedge \ldots \beta_{u_k,\mathbf{u}_k}(X)) \right)$.

Proposition 1. *For a DBN $B = (G, \Theta)$ over variables V with induced DTMC $M_B = (S, P, AP, L)$ whose transition relation is δ it holds that $\delta = \bigwedge_{v \in V} \delta_v$.*

3.2 Image Computation with Factored BDDs

Consider a Boolean formula $\Psi(X)$ that describes a set of states in the DTMC $M_B = (S, P, AP, L)$. The formula $\mathsf{Pre}(\Psi)(X) = \exists X'.\delta(X, X') \wedge \Psi(X')$ describes the set of states that have a successor in M_B which is a Ψ-state. The BDD describing $\mathsf{Pre}(\Psi)$ can be computed by applying the standard conjunction and existential abstraction operations to the BDDs for $\delta(X, X')$ and $\Psi(X')$.

When $\delta(X, X')$ is given in the factored form $\delta_{v_1}(X, X'), \ldots, \delta_{v_n}(X, X')$, where $V = \{v_1, \ldots, v_n\}$, we can avoid constructing the BDD for the global transition relation δ, and instead use the partitioned form in the computation of $\mathsf{Pre}(\Psi)$. Depending on the functions $\delta_{v_1}(X, X'), \ldots, \delta_{v_n}(X, X')$ and the number of variables on which each of them depends, their individual size can be much smaller than the size of δ. Furthermore, since each δ_v depends only on a subset

of X', it is possible that applying *early quantification* [6] can lead to avoiding a blow-up of intermediate results during the computation of $\mathsf{Pre}(\Psi)$. Such conjunctive partitioning of the transition relation has been successfully used for efficient forward-image computation in the verification of hardware models [6].

Here we describe the application of this approach to the Pre-image computation for the DBN $B = (G, \theta)$. Let $\pi : \{1, \ldots, n\} \to V$ be an ordering of the variables in V. We will explain later how to choose a potentially good ordering based on the dependency graph G of the DBN. For each $v \in V$, let $Y_v \subseteq X'$ be the set of variables in X' on which δ_v depends, that is $y \in Y_v$ iff $y \in X'$ and $\delta_v[0/y] \neq \delta_v[1/y]$, where $\delta_v[0/y]$ is the formula obtained from δ_v by substituting 0 for the variable y. Also, let $Z_v = Y_v \setminus \left(\bigcup_{i=\pi^{-1}(v)+1}^{n} Y_{\pi(i)}\right)$ be the set of variables in X' on which δ_v depends, but none of δ_u with $\pi^{-1}(u) > \pi^{-1}(v)$ depends on them. Note that the sets Z_v are pairwise disjoint. Then, $\mathsf{Pre}(\Psi)$ is computed by:

$$
\begin{aligned}
\Psi_1(X, X') &= \exists Z_{\pi(1)} \left(\delta_{\pi(1)}(X, X') \wedge \Psi(X')\right) \\
\Psi_2(X, X') &= \exists Z_{\pi(2)} \left(\delta_{\pi(2)}(X, X') \wedge \Psi_1(X, X')\right) \\
&\cdots \\
\Psi_n(X, X') &= \exists Z_{\pi(n)} \left(\delta_{\pi(n)}(X, X') \wedge \Psi_{n-1}(X, X')\right) \\
\mathsf{Pre}(\Psi)(X) &= \exists (X' \setminus (\bigcup_{v \in V} Z_v)) \Psi_n(X, X').
\end{aligned}
$$

The ordering π of the variables in X' is important, as it determines how many variables are existentially abstracted at each intermediate step, which can in turn influence the size of the intermediate BDDs. We now describe a heuristic that uses the dependency graph G to find a good ordering. Let G' be the restriction of G to the nodes V'. By traversing the graph G' in post-order we can compute π such that for every $i, j \in \{1, \ldots, n\}$, if $\pi(i) = v$, $\pi(j) = u$ and there is an edge in G' from u to v, then $i < j$. This allows for eliminating the variables X'_u at step j of the computation of $\mathsf{Pre}(\Psi)$, as none of the transition relations $\delta_{\pi(k)}$ considered at the subsequent steps $k > i$ depends on X'_u. Additionally, if variables $u \in V'$ and $v \in V'$ are mutually unreachable in G' but $|Y_u| < |Y_v|$, then δ_v will appear earlier in the ordering, leading to the elimination of more variables.

3.3 Reachability Computation

As we recalled in Section 2, the sets $\mathsf{Sat}(\mathbb{P}_{=0}(\Phi_1 \,\mathcal{U}\, \Phi_2))$ and $\mathsf{Sat}(\mathbb{P}_{=1}(\Phi_1 \,\mathcal{U}\, \Phi_2))$ can be computed by backward graph reachability starting from $\mathsf{Sat}(\Phi_2)$ in the (possibly modified) graph G_{M_B}. Here we show how to do that using the factored symbolic representation of the edge relation in G_{M_B}, constructed as above.

As usual, $\mathsf{Sat}(\mathbb{P}_{>0}(\Phi_1 \,\mathcal{U}\, \Phi_2))$ is computed as the least fixpoint $\mu U.\mathsf{Sat}(\Phi_2) \vee (\mathsf{Pre}(U) \wedge \mathsf{Sat}(\Phi_1))$, which corresponds to computing the states backward reachable from $\mathsf{Sat}(\Phi_2)$ that are in $\mathsf{Sat}(\Phi_1)$. For the computation of $\mathsf{Sat}(\mathbb{P}_{=1}(\Phi_1 \,\mathcal{U}\, \Phi_2))$, instead of restricting the transition relations δ_v to the set $(S \setminus (\mathsf{Sat}(\Phi_1) \cup \mathsf{Sat}(\Phi_2))$

in order to represent the transition relation of the modified graph, we use the following fixpoint expressions with the unmodified partitioned transition relation:

$$\Psi(X) = \neg\mu U.\mathsf{Sat}(\Phi_2) \vee (\mathsf{Pre}(U) \wedge \mathsf{Sat}(\Phi_1) \wedge \neg\mathsf{Sat}(\Phi_2))\,,$$
$$\mathsf{Sat}(\mathbb{P}_{=1}(\Phi_1\,\mathcal{U}\,\Phi_2)) = \neg\mu U.\Psi \vee (\mathsf{Pre}(U) \wedge \mathsf{Sat}(\Phi_1) \wedge \neg\mathsf{Sat}(\Phi_2))\,.$$

Next and bounded until formulas are handled in a similar way using the preimage computation based on the factored transition relation.

4 Model Checking Quantitative PCTL

4.1 Factored MTBDD Representation

Let $B = (G, \Theta)$ be a DBN over a set of finite-state variables V, and let the sets of variables $X = \bigcup_{v\in V} X_v$ and $X' = \bigcup_{v\in V} X'_v$ be as in the previous section. The transition probability matrix P of the induced DTMC $M_B = (S, P, AP, L)$ can be represented as an MTBDD ρ over the variables $X \cup X'$. Here we use again the structure of the DBN and the local dependencies implied by it to give a factored representation of ρ as the element-wise product of matrices ρ_v for $v \in V$.

As before, if $Pa_B(v') = \{u_1, \ldots, u_k\}$, we use $(\mathbf{v}, \mathbf{u}_1, \ldots, \mathbf{u}_k, p) \in \Theta_{v'|Pa_B(v')}$ to denote the elements of the conditional probability table $\Theta_{v'|Pa_B(v')}$ for v'.

For each $v \in V$ and $\mathbf{v} \in \mathit{Val}(v)$ (respectively $v' \in V'$ and $\mathbf{v} \in \mathit{Val}(v')$), we denote with $\mu_{v,\mathbf{v}}(X)$ (respectively $\mu_{v',\mathbf{v}}(X')$) the MTBDD for the Boolean formula equivalent to the atomic predicate $v = \mathbf{v}$ (respectively $v' = \mathbf{v}$). For $p \in \mathbb{R}$, we denote with μ_p the MTBDD that maps each assignment to $X \cup X'$ to the constant p. For each $v \in V$, we define the MTBDD $\rho_v(X, X') = \sum_{\mathbf{v}\in \mathit{Val}(v')} (\mu_{v',\mathbf{v}}(X') * \psi_{v,\mathbf{v}})$, where $\psi_{v,\mathbf{v}} = \sum_{\substack{(\mathbf{v},\mathbf{u}_1,\ldots,\mathbf{u}_k,p)\in\Theta_{v|Pa_B(v)}\\ p>0}} (\mu_p * \mu_{u_1,\mathbf{u}_1}(X) * \ldots * \mu_{u_k,\mathbf{u}_k}(X))$, and where $+$ and $*$ denote respectively sum and multiplication of real-valued functions represented as MTBDDs. Each ρ_v represents a real matrix whose rows are indexed by $\mathit{Val}(X)$, and whose columns are indexed by $\mathit{Val}(X')$. The matrix ρ_v describes the local dependency of the variables X'_v on the remaining variables. The transition probability matrix of M_B is obtained by taking the element-wise product of the transition probability matrices for the individual variables.

Proposition 2. *For a DBN $B = (G, \Theta)$ over variables $V = \{v_1, \ldots, v_n\}$ with induced DTMC $M_B = (S, P, AP, L)$ whose transition probability function is ρ it holds that $\rho = \rho_{v_1} * \ldots * \rho_{v_n}$.*

4.2 Matrix-Vector Multiplication with Factored MTBDDs

Let $A(X, X')$ be an MTBDD representing a square real matrix such that $\mathit{Val}(X)$ are the row indices and $\mathit{Val}(X')$ are the column indices. Let $b(X)$ be an MTBDD representing a real vector with indices $\mathit{Val}(X')$. The matrix-vector product $c = Ab$ can be computed symbolically [1] as $c(X) = \exists X'.A(X, X') * b(X')$, where

$*$ is the multiplication operation for real-valued functions (MTBDDs) and $\exists$ is the sum-abstraction. In our case, the transition probability matrix is given in factored form $\rho = \rho_{v_1} * \ldots * \rho_{v_n}$. Since the element-wise multiplication $*$ is associative and commutative, we can perform matrix-vector multiplication without computing ρ upfront as $\exists X'.\rho_{v_1}(X, X') * \ldots * \rho_{v_n}(X, X') * b(X')$.

Furthermore, as in the previous section we can employ early quantification whenever possible, trying to reduce the size of intermediate MTBDDs.

As we will see in our experimental results in Section 5, however, in the majority of the cases early quantification does not reduce the size of the intermediate MTBDDs. Although it might reduce the number of variables the function depends on, existential abstraction may increase the number of terminal nodes, thus affecting the amount of sharing between subgraphs of the MTBDD.

4.3 Solving Linear Equations

As we recalled in Section 2, model checking quantitative PCTL reduces to computing the satisfaction sets for qualitative formulas and solving systems of linear equations with real coefficients. Usually, symbolic methods based on MTBDDs for solving such systems employ iterative methods [16], since those do not require modifying the matrix during the computation, which is important for the compactness of the MTBDD representation. In order to seamlessly use the factored representation of the matrix, we use the power method, which only requires matrix-vector multiplication operations with the transition probability matrix.

As in the qualitative case, for until and bounded until formulas, instead of restricting each of the factors to the set $S_?$, which can introduce dependency on additional variables, we apply the restriction to the candidate solution vector at each step. We let $c^{(0)}(s) = 1$ for $s \in S_{=1}$ and $c^{(0)}(s) = 0$ for $s \in S_{=0} \cup S_?$, and then iteratively compute $c^{(i+1)} = Ac^{(i)} + b$, where at each step we modify $c^{(i)}$ according to S_1 and S_0. The power method receives as an input parameter a real value ε, and the iteration terminates when $||c^{(i+1)} - c^{(i)}||_\infty < \varepsilon$. As the power method is guaranteed to converge [3], we can compute an approximation to the solution vector up to a theoretically arbitrary precision. Using the power method based on partitioned transition probability matrix we compute $\mathsf{Sat}(\mathbb{P}_J(\Phi_1 \,\mathcal{U}\, \Phi_2))$. The method applies the matrix-vector multiplication procedure we described, using the ordering π to determine the order of applications of existential abstraction. Next properties are handled directly using the matrix-vector multiplication procedure for factored MTBDDs, and bounded until formulas are handled analogously to unbounded until formulas as described in Section 2.

5 Experimental Evaluation

We evaluate our approach on a set of several benchmarks. We have implemented a prototype PCTL model checker based on factored symbolic representations. Our tool is implemented in C++ using version 2.5.0 of the CUDD library [18]. In order to compare the performance of our technique to classical symbolic PCTL

Table 1. Model (MTBDD) size and peak BDD size for the verification of the respective qualitative property for several instances of each of the considered models.

	Model MTBDD size			Peak BDD size	
	Partitioned	Global	PRISM	Partitioned	Global
Linear N=10	7 (for $i = 1$); 10 (for $i > 1$)	1111	1111	11	29
Linear N=20	7 (for $i = 1$); 10 (for $i > 1$)	7816	7816	21	59
Linear N=30	7 (for $i = 1$); 10 (for $i > 1$)	25121	25121	31	89
Herman N=15	10 (for $i = 1$); 8 (for $i > 1$)	810	810	626	3090
Herman N=17	10 (for $i = 1$); 8 (for $i > 1$)	1053	1053	927	4704
Herman N=19	10 (for $i = 1$); 8 (for $i > 1$)	1328	1328	1377	6672
Herman N=21	10 (for $i = 1$); 8 (for $i > 1$)	1635	1635	1730	8810
Tree L=4	9	3547	3688	44	193
Tree L=5	9	178855	185884	92	3549
Tree L=6	9	TO	MO	188	
Sensor K=4	44	75206	83885	511	21191
Sensor K=5	44	TO	MO	1436	
Sensor K=6	44	TO	MO	4300	

model checking we also implemented all procedures using monolithic symbolic representation.[1] We also compare to the state-of-the-art symbolic probabilistic model checker PRISM [13], version 4.3.

For the comparison with PRISM, we use options `-m -cuddmaxmem 2g`, that is, the symbolic engine with memory limit for CUDD increased to 2GB. Each experiment was run on a 3 GHz Intel Xeon processor with a timeout of 10 hours.

We consider several probabilistic systems that can be naturally modelled as DBNs, and which exhibit different structure of their underlying dependency graphs. In our first example this graph has a simple linear structure, and thus, there exists a natural variable ordering, in which variables that directly depend on each other are close. The canonical Herman's protocol benchmark [11], which we consider next, also falls into this category. As an instance with a more complex dependency structure we then consider a network model where nodes are organized in a full binary tree. Such dependency structure arises commonly in fault tree analysis [17]. We also consider an instance with a grid structure, as an abstraction of device networks such as sensor and communication grids [9].

Now we describe the benchmarks and give a summary of our experimental results. Then, in subsection 5.1 we interpret and discuss these results.

Network with a linear topology. As our first benchmark we consider a network of N computers organized in a simple linear topology: for each $i \in \{1, \ldots, N-1\}$ there is an unidirectional connection from machine i to machine $i+1$. Each machine is associated with a Boolean variable up_i, which indicates whether at the current step the machine is up or down. A machine which is up can fail with probability p in the next step. A machine $i > 1$ which is down, can be rebooted with probability q, only if machine $i-1$ is up in the current step. This defines conditional probability distributions $Pr(up'_i \mid up_i, up_{i-1})$ for $i = 2, \ldots, N$.

We used values $p = q = 0.4$ and considered the following verification tasks:

(1) The property $\mathbb{P}_{=1}(\Diamond$ "all machines are down") holds in every state.

(2) Compute the probability of $\Diamond^{\leq 10}$ "machine N is down", for the initial state in which all machines are up.

[1] The code is available at `http://www.mpi-sws.org/~rayna/atva16-experiments/`

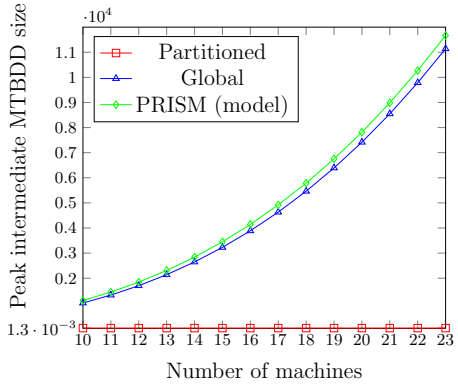

Fig. 1. Results for verification task (2) for the linear topology benchmark.

(3) Compute the probability of $\Diamond^{\leq 10}$ "exactly one machine is up", for the initial state in which all machines are up.

The sizes of the MTBDDs for the partitioned and the global transition relations for $N \in \{10, 20, 30\}$ are shown in Table 1. There we also show the peak BDD size reached during the verification of the qualitative property (1). The table also contains these results for selected instances of the other benchmarks.

Figure 1 shows a comparison of the peak MTBDD size reached during verification task (2) executed for $N = 10, \ldots, 23$. For this specific quantitative property the peak size when using the factored representation remains constant, and when using the monolithic transition relation grows. This is not the case for verification task (3), as it can be seen from Figure 2, which shows that for the respective property the peak MTBDD sizes are essentially equal for the two approaches. For $N = 23$ in verification task (3) our approach runs out of memory, and the classical algorithm based on the global transition relation exceeds the time limit of 10 hours at $N = 19$. Regarding the execution time, while for verification task (2) all instances complete in under 0.1 seconds, for (3) we see in Figure 2 that our approach has better performance.

Since for verification task (2) the peak size of the MTBDD does not increase with N, our approach can verify this property even in cases when the MTBDD for the global transition system cannot be constructed. This is the case for example for $N = 100, 200, 300$, where our approach completes successfully, but PRISM exceeds even a 20 GB memory limit while building the model.

Herman's self-stabilization protocol. Herman's protocol [11] is a distributed self-stabilization algorithm. This is a randomized protocol for N processes (where N is odd) organized in a ring communication topology to reach a stable state, in which exactly one process has a token. Each process is associated with a random Boolean variable x_i, and process i has a token if $x_i = x_{((i-1) \mod N)}$. If process i has a token it sets x_i to 0 or 1, each with probability $\frac{1}{2}$. Otherwise, it sets x_i to $x_{((i-1) \mod N)}$. This defines conditional probability distributions $Pr(x_i' \mid x_i, x_{((i-1) \mod N)})$ for $i = 1, \ldots, N$. We consider the following properties:

(1) Every state satisfies the property $\Phi_1 = \mathbb{P}_{=1}(\Diamond \mathit{stable})$.

(2) Every state satisfies the property $\Phi_2 = \mathbb{P}_{\geq \frac{1}{2}}(\Diamond^{\leq hN^2} \mathit{stable})$, where hN^2 with $h = \frac{4}{27}$ is the upper bound on the expected stabilization time [4].

The MTBDD sizes for the partitioned and the global transition relations are shown again in Table 1 for $N \in \{15, 17, 19, 21\}$, as well as the peak BDD size for the qualitative property. Regarding the peak MTBDD sizes, the situation is similar to property (3) in the linear topology case: we do not observe a significant difference between the partitioned and the global versions. For $N = 15$ we have 25179 nodes in both cases, increasing to 400370 (respectively 401543) for $N = 19$. For $N = 21$ the partitioned approach runs out of memory, while the global one exceeds the timeout (PRISM successfully verified the property, in more than 11 hours). Here, our approach does not exhibit significantly better running time.

Network with a tree topology. Next we consider a network of machines organized in a full binary tree with L levels, consisting of $2^L - 1$ machines. Again, each machine i can be up or down and is associated with a Boolean random variable up_i. A machine at a leaf node can at each step be down with probability p and up with probability $1 - p$. A machine i at a non-leaf node is only in use if both of its children, machines $2i + 1$ and $2i + 2$ are down, and can only then fail, again with probability p. In our experiments we let $p = 0.6$ and analyze the probability of the system going down (i.e., the machine at the root going down):

(1) The property $\mathbb{P}_{=1}(\Diamond$ "the root is down") holds in every state.

(2) Compute the probability of $\Diamond^{\leq 1000}$ "the root is down", for the initial state in which all machines are up.

We show the results for the quantitative case for trees with $L = 4, \ldots, 10$ levels in Figure 3. Here we observe a significant difference in terms of the peak intermediate MTBDD size. For $L = 5$ the factored representation results in more than 60 times smaller peak MTBDD, and for $L = 6$ the global approach reaches the time limit, while PRISM runs out of memory (given 20 GB of memory PRISM also runs past the 10 hour mark). Our approach does not reach the timeout even for the tree with 10 levels, as shown in the right plot in Figure 3.

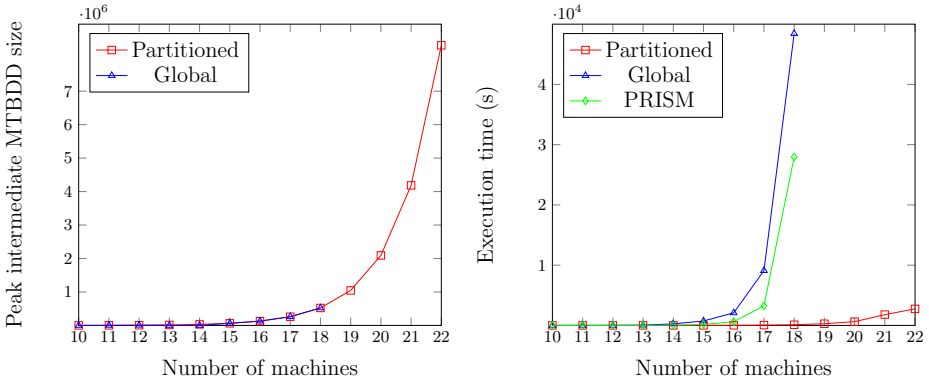

Fig. 2. Results for verification task (3) for the linear topology benchmark.

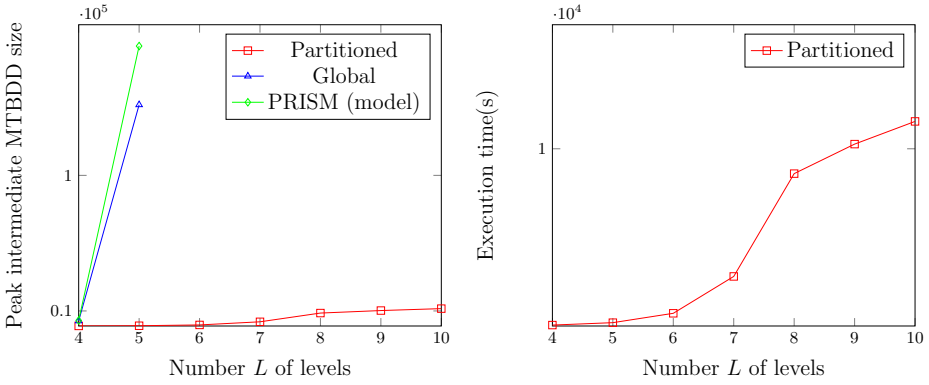

Fig. 3. Results for verification task (2) for the tree topology benchmark.

Sensor grid. Our final benchmark is a sensor network model described in [9]. We consider a simplified setting where we do not model power consumption and lost nodes. The sensor network models a forest fire alarm system which consists of $K \times K$ sensors organized in a grid. The purpose of the system is to detect fires and carry the alerts to the boundary of the forest (grid), while each sensor communicates directly only with its four neighbours. At each time point each sensor is in one of four possible states: sleep, sense, listen and broadcast. From the sleep state a sensor goes with probability $\frac{1}{2}$ to sense and with probability $\frac{1}{2}$ to listen. If a sensor in state sense detects fire it goes to state broadcast and stays there forever. Otherwise it goes to state listen where it checks if one of its neighbours is broadcasting. If this is the case, it starts broadcasting in state broadcast forever, and otherwise it goes back to state sleep. In the initial state there is a fire at a fixed single cell (F_x, F_y). We analyze the probability of reaching a state in which there is fire still only at (F_x, F_y) (i.e., we assume that the fire does not spread), and the alarm is successfully propagated to the grid's boundary:

(1) Verify the property that if there is a fire only at a fixed cell (F_x, F_y), then with probability 1 the alarm reaches the boundary.

(2) Compute the probability that if there is a fire only at a fixed cell (F_x, F_y), then the alarm reaches the boundary within b time steps.

We consider grids of size $K \in \{4, 5, 6\}$ and fix $(F_x, F_y) = (2, 2)$. For $K = 4$ we set $b = 20$, for $K = 5$ and $K = 6$ we set $b = 5$. For $K = 4$, the peak MTBDD size for our approach is 3783, while for the verification using the global transition relation this is 69893 (the size of the global MTBDD is actually larger and is 75206) and the size of the PRISM model MTBDD is 83885. For $K = 5$ we have peak size of 6718952, while the construction of the global MTBDD times out, and PRISM runs out of memory during the model construction.

5.1 Discussion

The objective of this work is to evaluate factored symbolic representations for quantitative verification. We observe that the use of the partitioned symbolic representation leads to negligible or no improvement in the majority of cases.

As it can be expected, the size of the model representation is considerably smaller in the partitioned version, and for *qualitative* properties the advantages of partitioned representation and early quantification do carry over from non-probabilistic verification. Unfortunately, the same cannot be said about *quantitative* verification. The reason is that early variable elimination can decrease the number of variables, but not the number of different terminals in an MTBDD. For example, for Herman's protocol with 15 processes we observe multiplication operations where the maximum number of terminals reached during the sequence of early variable elimination steps is 12362, while the result of the multiplication has 612 terminals. For the same operation, with the same end result, the intermediate result when using the global transition matrix has 2880 terminals.

For very simple quantitative properties and systems, such as property (2) in the linear topology benchmark, which only refers to the last machine in the linear network, we do observe notable effects on the peak MTBDD size. However, in these cases already the classical symbolic verification methods are quite efficient, and the performance improvement is not dramatic.

For quantitative properties that refer to all the variables in the DBN or where intermediate verification steps require more global reasoning, the partitioned approach does not perform better than the standard one. Indeed, for property (3) in the linear topology benchmark and the quantitative stabilization property for Herman's protocol the peak MTBDD size using the factored representation is comparable to the size of the MTBDD for the global transition relation.

These observations extend also to the benchmarks with more complex dependence graphs. While factorization is beneficial for the simple property that refers only to the root node of the tree, the analysis of the sensor network benchmark is prohibitively hard even for the partitioned approach. For size of the grid $K = 5$ our method succeeds, reaching a peak MTBDD size of more than 6 million nodes, while the other two run out of time and memory respectively, during model construction. However, already for $K = 6$ our approach exceeds the timeout as well, and thus also does not scale beyond small values of K.

6 Conclusion

We presented and evaluated a symbolic model checking approach based on a partitioned symbolic representation of Markov chains induced by DBNs. Our experimental results indicate that known techniques for exploiting model structure in symbolic verification are not efficient in the quantitative setting. While factorization proves to be efficient for model checking qualitative PCTL properties, we conclude that achieving scalability in quantitative reasoning for DBNs requires exploring different avenues.

References

1. R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their ap-

plications. In *Proceedings of the 1993 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '93, pages 188–191, 1993.
2. Christel Baier, Edmund M. Clarke, Vasiliki Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan. Symbolic model checking for probabilistic processes. In *Proc. ICALP'97*, volume 1256 of *LNCS*, pages 430–440. Springer, 1997.
3. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.
4. Maria Bruna, Radu Grigore, Stefan Kiefer, Joël Ouaknine, and James Worrell. Proving the Herman-protocol conjecture. *CoRR*, abs/1504.01130, 2015.
5. Randal Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.
6. J. R. Burch, E. M. Clarke, and D. E. Long. Representing circuits more efficiently in symbolic model checking. In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, DAC '91, pages 403–407. ACM, 1991.
7. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 1020 states and beyond. *Inf. Comput.*, 98(2):142–170, June 1992.
8. Edmund M. Clarke, Kenneth L. McMillan, Xudong Zhao, Masahiro Fujita, and J. Yang. Spectral transforms for large Boolean functions with applications to technology mapping. In *DAC*, pages 54–60, 1993.
9. A. Demaille, S. Peyronnet, and B. Sigoure. Modeling of sensor networks using XRM. In *Leveraging Applications of Formal Methods, Verification and Validation, 2006. ISoLA 2006. Second International Symposium on*, pages 271–276, Nov 2006.
10. M. Fujita, P.C. McGeer, and J.C.-Y. Yang. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2):149–169, 1997.
11. T. Herman. Probabilistic self-stabilization. *Information Processing Letters*, 35(2):63–67, 1990.
12. Holger Hermanns, Joachim Meyer-Kayser, and Markus Siegle. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In *3rd International Workshop on the Numerical Solutions of Markov Chains, NSMC 99*, pages 188–207, 1999.
13. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.
14. Christopher J Langmead, Sumit Kumar Jha, and Edmund M Clarke. Temporal-logics as query languages for dynamic Bayesian networks: Application to d. melanogaster embryo development. 2006.
15. Sucheendra K. Palaniappan and P. S. Thiagarajan. Dynamic Bayesian networks: A factored model of probabilistic dynamics. In *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, volume 7561 of *LNCS*, pages 17–25. Springer, 2012.
16. D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.
17. Enno Ruijters and Mariëlle Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15:29–62, 2015.
18. Fabio Somenzi. CUDD: BDD package, University of Colorado, Boulder. `http://vlsi.colorado.edu/~fabio/CUDD/`.