Journal of Artificial Intelligence Research 77 (2023) 1637-1696 Submitted 07/2022; published 08/2023

Automatically Finding the Right Probabilities in Bayesian Networks

Bahare Salmani SALMANI@CS.RWTH-AACHEN.DE
Joost-Pieter Katoen KATOEN@CS.RWTH-AACHEN.DE
RWTH Aachen University
Aachen, Germany

Abstract

This paper presents alternative techniques for inference on classical Bayesian networks in which all probabilities are fixed, and for synthesis problems when conditional probability tables (CPTs) in such networks contain symbolic parameters rather than concrete probabilities. The key idea is to exploit probabilistic model checking as well as its recent extension to parameter synthesis techniques for parametric Markov chains. To enable this, the Bayesian networks are transformed into Markov chains and their objectives are mapped onto probabilistic temporal logic formulas.

For exact inference, we compare probabilistic model checking to weighted model counting on various Bayesian network benchmarks. We contrast symbolic model checking using multi-terminal binary (aka: algebraic) decision diagrams to symbolic inference using probabilistic sentential decision diagrams, symbolic data structures that are tailored to Bayesian networks.

For the parametric setting, we describe how our techniques can be used for various synthesis problems such as computing sensitivity functions (and values), simple and difference parameter tuning and ratio parameter tuning. Our parameter synthesis techniques are applicable to arbitrarily many, possibly dependent, parameters that may occur in multiple CPTs. This lifts restrictions, e.g., on the number of parametrized CPTs, or on parameter dependencies between several CPTs, that exist in the literature. Experiments on several benchmarks show that our parameter synthesis techniques can treat parameter synthesis for Bayesian networks (with hundreds of unknown parameters) that are out of reach for existing techniques.

1. Introduction

Bayesian networks (BNs, Darwiche, 2009) are a popular model in AI and decision making. Their graphical nature makes them relatively easy to comprehend and enables the usage of efficient graph algorithms, e.g., for determining conditional (in)dependencies between (sets of) random variables. Moreover, they provide a succinct representation of complex joint probability distribution functions. Efficient algorithms e.g., (Darwiche, 2002; Sang et al., 2005; Chavira & Darwiche, 2005, 2006, 2008; Bart et al., 2016) have been developed to analyse BNs, most notably to perform inference, a problem whose theoretical complexity is PP-complete (Cooper, 1990; Dagum & Luby, 1993). Dedicated data structures such as variants of decision diagrams (Fujita et al., 1997; Bahar et al., 1997; Sanner & McAllester, 2005; Kisa et al., 2014) can compactly represent BNs enabling fast inference. Conditional probability tables (CPTs) in BNs define a probability distribution over the possible values of a random variable conditioned on the values of its parents in the graph structure.

Variants of BNs have been considered where CPT entries are symbolic expressions over a fixed set of parameters (such as $p, 1{-}p$, and $1{-}p{-}q$, etc. with $0 < p, q < 1$) rather than constants. Such *parametric* BNs are very useful in situations where probabilities are unknown, or partially known. They received a lot of attention, see e.g., (Coupé & Van der Gaag, 1998; Coupé et al., 2000; Druzdzel & Van der Gaag, 2000; Jensen, 1999; Laskey, 1995; Castillo et al., 1995, 1996, 1997a; Kjærulff & Van der Gaag, 2000; Chan & Darwiche, 2002, 2004; Coupé & Van der Gaag, 2002; Renooij, 2014; Bolt & Van der Gaag, 2015). Important objectives on parametric BNs are e.g., to provide a symbolic expression in terms of the model parameters p and q for inference queries, sensitivity analysis, or determining whether there exists a concrete set of parameter values such that a given threshold on an inference query holds. A relevant objective is parameter tuning: find the minimal change of parameters such that some constraint, e.g., $\Pr(H{=}h \mid E{=}e) > q$ holds (Chan & Darwiche, 2005; Laskey, 1995).

As sensitivity analysis and parameter synthesis are computationally hard in general (Laskey, 1995; Kjærulff & Van der Gaag, 2000; Kwisthout & Van der Gaag, 2008), many existing techniques restrict the number of parameters per CPT (n-way for small n e.g., by Chan & Darwiche, 2002; Coupé & Van der Gaag, 2002; Van der Gaag, Renooij, & Coupé, 2007), forbid parameter dependencies in several CPTs (single CPT by Chan & Darwiche, 2004), or consider specific structures such as join trees (Kjærulff & Van der Gaag, 2000) and require all parameters to occur in the same clique of the junction tree.

This paper presents a new approach to analyze BNs and their parametric variant. The key idea of our approach is to exploit techniques from the field of probabilistic model checking (Baier & Katoen, 2008; Katoen, 2016; Baier, de Alfaro, Forejt, & Kwiatkowska, 2018). Probabilistic model checking (PMC) is a fully automated verification technique for Markov models that has been successfully applied in various application domains. It checks whether a given property, expressed in temporal logic, holds with a given probability. Its core problem is to compute reachability probabilities: is the probability to reach a certain set of target states from the initial state below or above a certain threshold? Efficient algorithms exist to compute such probabilities, and—similar to the case for BNs—variants of decision diagrams have been used to compactly store Markov models and efficiently manipulate them. Most modern PMC tools (Kwiatkowska et al., 2011; Dehnert et al., 2017) have a mode in which they exploit such decision diagrams.

Significant progress has been recently made in the automated analysis of parametric Markov models, models in which the transition probabilities are multivariate polynomials over a fixed set of parameters such as p and $1{-}p$. Substitution of these parameters by concrete values induces a probability distribution over the state space of the MC. About 15 years ago, the techniques focused on computing a rational function over the parameters expressing the reachability probability of a given target state (Daws, 2004; Lanotte et al., 2007; Hahn et al., 2011) and could handle only up to a handful of parameters. In recent years significant progress has been made to check whether there exists a parameter valuation inducing an MC that satisfies a given objective, or to partition the parameter space — the space of all

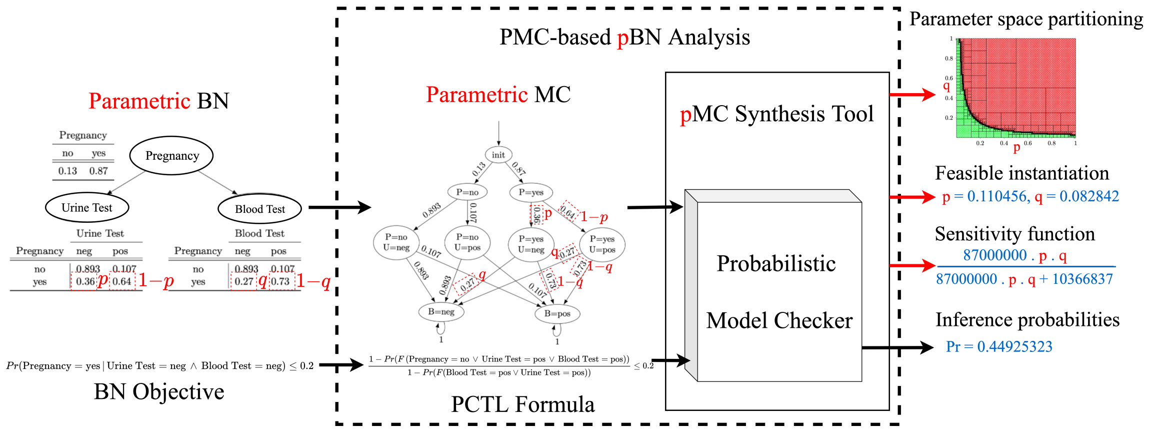


Figure 1: Overview of our framework for analyzing (parametric) Bayesian networks.

possible parameter values — into "good" and "bad" w.r.t. a given objective, e.g., is the probability to reach some states below (or above) a given threshold λ? Due to algorithmic improvements, nowadays Markov models with hundreds of thousands of states and tens or hundreds of parameters are in reach (Dehnert et al., 2015; Quatmann et al., 2016; Gainer et al., 2018; Fang et al., 2021; Heck et al., 2022); for a recent overview see (Junges et al., 2019).

This paper applies state-of-the-art probabilistic model checking and parameter synthesis to various analysis problems on (parametric) Bayesian networks. A bird eye's view on our approach is provided in Fig. 1. (Parametric) BNs are mapped automatically onto (parametric) Markov chains (MCs) and temporal logic formulas in PCTL (Hansson & Jonsson, 1994) are associated to the objectives at hand on (p)BNs. PMC and parameter synthesis techniques are then employed to obtain various sorts of information on the (p)BNs such as probabilities for inference queries, sensitivity functions for a given evidence in terms of the parameters, parameter instantiations that make certain constraints true, or even partitioning of the entire (in the figure a two-dimensional) parameter space.

Organization and contributions of the paper. This paper provides details on all ingredients of our approach. We start off in Section 2 by describing the models of interest: parametric BNs and parametric MCs. This is complemented by treating symbolic techniques on both models: probabilistic sentential decision diagrams (PSDDs) (Choi, Kisa, & Darwiche, 2013; Kisa et al., 2014), a symbolic data structure tailored to BN analysis, and multi-terminal (aka: algebraic) decision diagrams (MTBBDs) (Fujita et al., 1997; Bahar et al., 1997) that are used in e.g., symbolic PMC (Baier, Clarke, Hartonas-Garmhausen, Kwiatkowska, & Ryan, 1997) as well as in Bayesian inference (Sanner & McAllester, 2005).

Section 3 defines a variety of objectives on BN, ranging from classical threshold inference problems to more advanced parameter tuning objectives. This is complemented by several objectives on parametric MCs.

Section 4 presents two techniques for mapping a BN onto a corresponding MC: a general translation that is applicable to all objectives on BNs, and a translation that takes the evidence of the BN query into account and tailors the MC to this specific query. For a range of objectives on (parametric) BNs, the corresponding temporal logic formulas are provided.

Section 5 surveys the main PMC algorithms and relevant model reduction techniques. We then report on various experiments for analyzing classical, i.e., non-parametric BNs in Section 6. An extensive experimental validation using a prototypical implementation on top of the Storm model checker (Dehnert et al., 2017) compares the efficiency of probabilistic inference using explicit-state PMC techniques to Ace, a tool that carries out inference on BNs using compilation into arithmetic circuits. In addition, symbolic probabilistic inference using MTBDD-based PMC is compared to PSDD-based inference on BNs. We also investigate the effect of the evidence-tailored translation from BNs onto MCs.

Section 7 surveys algorithms for various parameter synthesis objectives on MCs, such as computing solution functions, feasibility checking (find a parameter instantiation, if any, such that a given constraint holds), parameter space partitioning (divide the parameter space into accepting and refuting regions), and region verification (does a given constraint hold for a given set of parameter instantiations?). All algorithms in this paper are described at an intuitive level, and some are illustrated with examples; references to detailed descriptions of the algorithms are provided.

Section 8 reports on the experimental validation of our approach using a tool-chain built on top of the Storm model checker as well as the parameter synthesis tool Prophesy (Dehnert et al., 2015). The tools Bayesserver and SamIam are used as baseline tools for evaluating parametric BNs. The experiments focus on the following issues:

- Can we perform sensitivity analysis on parametric BNs fast and with arbitrary parameter dependencies?
- What are the decisive factors in the computation time of pBN sensitivity functions using PMC techniques?
- How effective are the parametric MC feasibility checking techniques for analyzing parametric BNs?
- How do the number of parameters influence the feasibility analysis time?
- How does the coverage affect the parameter space partitioning time for pBNs?

Section 9 discusses related work that has not yet been discussed in the rest of the paper and Section 10 concludes the paper and provides some pointers to future work. Appendix A contains the proofs of the main theoretical results.

This paper extends the conference papers (Salmani & Katoen, 2020, 2021). It has thoroughly been rewritten and has been extended with a more detailed description of the objectives on pBNs and pMCs, and with a complete description of all relevant algorithms for PMC and parameter synthesis. All experiments have been newly conducted, are more extensive, and include new techniques for feasibility checking using gradient descent as well as a comparison between three algorithms for feasibility checking.

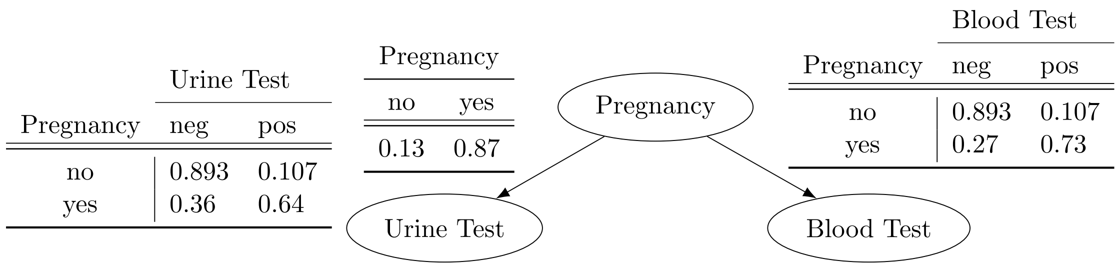
The main findings of this paper. In a nutshell, our main findings are:

- Inference queries on (p)BNs correspond to reachability probabilities in (p)MCs.
- Temporal logic yields a flexible technique for formalizing inference queries.
- Probabilistic inference using model checking is competitive except if many probability values frequently occur in CPTs.
- Inference using PSDD outperforms inference using MTBDD-based model checking.
- Evidence-tailored compilation can reduce the size of decision diagrams by up to one order of magnitude.
- Current techniques for computing solution functions in PMC scale much better than existing techniques for parametric BNs: pBNs with hundreds of parameters can be analysed in reasonable time.
- Parameter synthesis technique from PMC can deal with multiple parameters within a BN, possibly occurring in different CPTs.
- Gradient descent is mostly the favorable technique for finding a suitable parameter instantiation and can deal with parametric BNs with up to hundreds of parameters. For parametric BNs with many (more than 200) parameters, convex programming techniques are more efficient.
- Parameter space partitioning is very effective for parameter tuning for parametric BNs with up to ten parameters.

2. Parametric Probabilistic Models

After providing some preliminaries on parameters and their valuations, this section formalises the notions of parametric Bayesian networks and parametric Markov chains. It finishes with describing two symbolic data structures to obtain succinct representations of probabilistic models: multi-terminal binary (aka algebraic) decision diagrams, and probabilistic sentential decision diagrams.

Variables. Let V be a set of m random variables. We fix a total order on the random variables in V by letting $V = \{v_1, \ldots, v_m\}$. Let D_{v_i} denote the domain of variable v_i. Let $\mathit{Eval}(V) = D_{v_1} \times \ldots \times D_{v_i}$, i.e., the Cartesian product of the domains, represents all possible ways to assign a joint valuation to the random variables. For convenience, each *variable valuation* $\eta \in \mathit{Eval}(V)$ will be represented as a map $\eta\colon V \to D_{v_1} \cup \ldots \cup D_{v_m}$. For example, let $V = \{U, B\}$ with $D_U = D_B = \{\mathit{neg}, \mathit{pos}\}$. The set of variable valuations is $\mathit{Eval}(V) = \{(\mathit{neg}, \mathit{neg}), (\mathit{neg}, \mathit{pos}), (\mathit{pos}, \mathit{neg}), (\mathit{pos}, \mathit{pos})\}$. *Parameters.* Let X be a set of n real-valued parameters $x_1, \ldots, x_n$ and let $\mathbb{Q}[X]$ denote the set of multivariate polynomials with rational coefficients over X. A *parameter instantiation* is a function $u : X \to \mathbb{R}$. We often consider u as a vector $\vec{u} \in \mathbb{R}^n$ by ordering the set of variables $X = \{x_1, \ldots, x_n\}$ and setting $u_i = u(x_i)$. We assume all parameters are bounded; i.e., $lb_i \leq u(x_i) \leq ub_i$ for each parameter x_i. Let $I_i = [lb_i, ub_i]$ denote the interval for the possible values of parameter x_i. The *parameter*



	Urine Test	
Pregnancy	neg	pos
no	0.893	0.107
yes	0.36	0.64

Pregnancy	
no	yes
0.13	0.87

	Blood Test	
Pregnancy	neg	pos
no	0.893	0.107
yes	0.27	0.73

Figure 2: An example BN - Pregnancy tests.

space of X, denoted by $\mathcal{U} \subseteq \mathbb{R}^n$, is the set of all possible parameter values, i.e., the hyper-rectangle spanned by the intervals $[lb_i, ub_i]$ for all i. A set $R \subseteq \mathcal{U}$ is called a *region*. A polynomial f can be seen as a function $f : \mathbb{R}^n \to \mathbb{R}$ where $f(u)$ is obtained by substituting every occurrence of x_i in f by $u(x_i)$; e.g., for $f = 2x_1^2 + x_2$, $u(x_1) = 3$ and $u(x_2) = 2$, we have $f(u) = 20$. From now on, we write $f[u]$ instead of $f(u)$ to make it clear when substitution occurs. Let $\mathit{Distr}(D)$ denote the set of parametric probability distributions over the finite domain D, i.e., the set of functions $\mu : D \to \mathbb{Q}[X]$. We thus consider finite-support parametric distributions that are (point-wise) multivariate polynomials with rational coefficients over the parameters. For all *well-defined* instantiations u, we have $0 \leq \mu(d)[u] \leq 1$ and $\Sigma_{d \in D}\, \mu(d)[u] = 1$. For example, let μ be the parametric probability distribution over the parameters $X = \{x\}$ and the domain $D = \{d, d', d''\}$ defined by $\mu(d) = x$ and $\mu(d') = 1{-}x$, and $\mu(d'') = 0$. For all instantiations u with $0 \leq u(x) \leq 1$, $\mu[u]$ is a distribution, e.g., for $u(x) = 1/4$, $\mu[u](d) = 1/4$ and $\mu[u](d') = 3/4$.

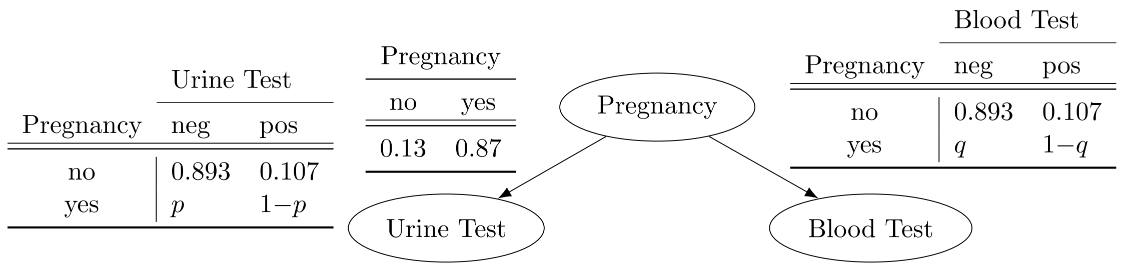
2.1 Parametric Bayesian Networks

A *parametric Bayesian network* is a Bayesian network in which some or all the entries in the conditional probability tables (CPTs) are polynomials over the parameters in X.

Definition 1. A *parametric BN* (pBN) $\mathcal{B}$ is a tuple (V, W, X, Θ), where $V = \{v_1, \ldots, v_m\}$ is a set of *discrete random variables*, $\mathcal{G} = (V, W)$ is a *directed acyclic graph* over the random variables V and the edges $W \subseteq V \times V$, $X = \{x_1, \ldots, x_n\}$ is a finite set of real-valued *parameters*, and $\Theta = \{\, \Theta_{v_i} \mid v_i \in V \,\}$ is a set of *parametric conditional probability tables* for the random variables in V.

Let $\mathit{parents}(v_i)$ denote the set of parents for the node v_i in $\mathcal{G}$. The CPT for variable v_i is the function $\Theta_{v_i} \colon \mathit{Eval}(\mathit{parents}(v_i)) \to \mathit{Distr}(D_{v_i})$ that maps each evaluation $\overline{par} \in \mathit{Eval}(\mathit{parents}(v_i))$ to a parametric probability distribution $\Theta_{v_i}(\overline{par})$ over D_{v_i}. The *CPT entry* $\Theta_{v_i}(\overline{par})(d_i)$ denotes the probability of $v_i = d_i$ given the parents evaluation $\overline{par}$.

A pBN without parameters, i.e., $X = \emptyset$, is an ordinary BN, provided that the CPTs Θ yield probability distributions over the domains of the vertices, i.e., if $\Theta_{v_i}(\overline{par})[u] \in \mathit{Distr}(D_{v_i})$ for each $v_i \in V$ and parent evaluation $\overline{par} \in \mathit{Eval}(\mathit{parents}(v_i))$. Let $\mathcal{B}[u]$ be obtained by replacing every parameter x_i in $\mathcal{B}$ by its value $u(x_i)$ where $u : X \to \mathbb{R}$. The parameter instantiation u is *well-defined* if $\mathcal{B}[u]$ is an ordinary BN. In the sequel, we assume u to be



Pregnancy		
no	yes	
0.13	0.87	

	Urine Test	
Pregnancy	neg	pos
no	0.893	0.107
yes	p	$1-p$

	Blood Test	
Pregnancy	neg	pos
no	0.893	0.107
yes	q	$1-q$

Figure 3: An example parametric BN - Pregnancy tests.

well-defined. A pBN defines a *parametric* joint probability distribution function (pdf) over V; similarly, the BN $\mathcal{B}[\emptyset]$ defines a joint pdf over V.

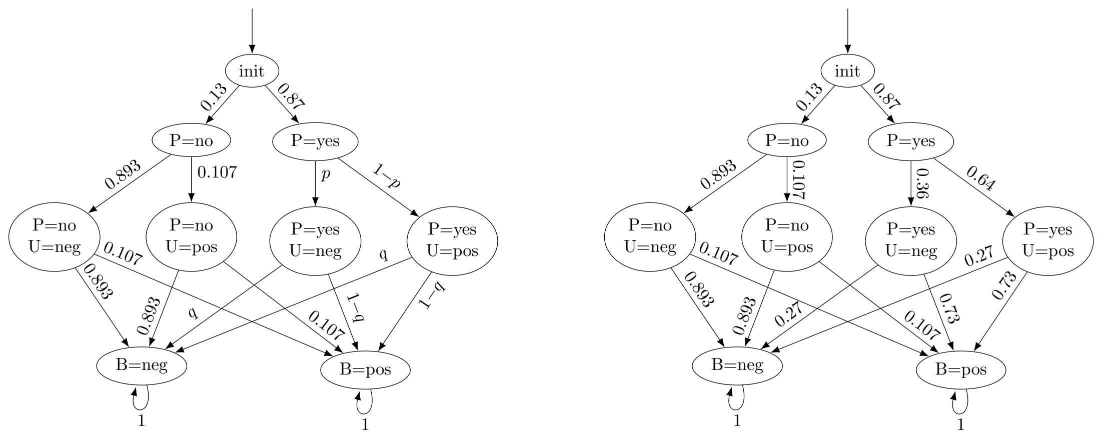
Example 1. Figure 2 depicts an ordinary BN example adopted from (Darwiche, 2009). The BN models the pregnancy of cows on a farm. The pregnancy can be tested by two types of tests: a urine test and a blood test. Figure 3 indicates a parametric version over the parameters $X = \{p, q\}$, where the *false negative* probabilities of both tests are unknown. The parameters p and q represent the *false negative* probability of the urine test and the blood test, respectively. The BN in Fig. 2 is obtained from the pBN in Fig. 3 by the well-defined instantiation u with $u(p) = 0.36$ and $u(q) = 0.27$.

Classes of parametric BNs. We define some classes of pBNs that are covered in existing sensitivity analysis techniques and tools. They constrain the number of parameters as well as the number of CPTs and the number of separate rows in a CPT containing parameters. Let $\mathcal{B}$ be a pBN over the set X of parameters. Let $c(x_i)$ be the number of CPTs in $\mathcal{B}$ in which x_i occurs and $r(x_i)$ denote the total number of rows that x_i occurs in across all CPTs.

- $\mathcal{B} \in p_1c_1r_1$ if and only if $\mathcal{B}$ contains one parameter, which only occurs in a single row of a single CPT, i.e., $X = \{x_1\}$, $c(x_1) = r(x_1) = 1$.
- $\mathcal{B} \in p_2c_{\leq 2}r_1$ if and only if $\mathcal{B}$ involves two parameters occurring only in a single row of two (or one) CPTs, i.e., $X = \{x_1, x_2\}$, $c(x_i) \in \{1, 2\}$, and $r(x_i) = 1$ for $i = 1, 2$.
- $\mathcal{B} \in p_*c_1r_1$ if and only if $\mathcal{B}$ allows arbitrarily many parameters, provided each parameter occurs in a single row of a single CPT, i.e., $r(x_i) = 1$, $c(x_i) = 1$ for each x_i. Each parameter x_i occurs in a separate row of the same CPT.

The class $p_1c_1r_1$ is used in one-way, $p_2c_{\leq 2}r_1$ in two-way sensitivity analysis (Chan & Darwiche, 2002; Coupé & Van der Gaag, 2002; Van der Gaag et al., 2007), and the class $p_*c_1r_1$ in single CPT (Chan & Darwiche, 2004). As we will see later on, our parameter synthesis techniques are applicable to the class $p_*c_*r_*$, i.e., pBNs with arbitrarily many parameters that can occur everywhere in the CPTs. Note that the definition also allows multiple distinct parameters to occur in the same CPT row and/or the same CPT entry.

Example 2. The pBN in Fig. 3 belongs to the class $p_2c_2r_1$ as it involves two parameters occurring in a single row in two CPTs. A variant of this example is depicted in Fig. 14(a)



(a) A pMC with unknown probabilities.

(b) An MC with fixed probabilities.

Figure 4: The *parametric* MC (a) and the MC (b) of the *pregnancy* example.

(page 1665) where the urine test is used twice for each cow to detect pregnancy. This pBN is in class $p_2c_2r_2$: parameter p occurs in two different rows of two different CPTs. This is an example with *parameter dependencies* as a parameter occurs in several CPTs.

2.2 Parametric Markov Chains

Whereas pBNs generalize BNs by allowing polynomials over parameters in CPTs, parametric Markov chains allow such polynomials as transition labels. Formally,

Definition 2. A *parametric Markov chain* (pMC) $\mathcal{M}$ is a tuple $(S, s_I, X, \mathcal{P})$ where S is a *finite* set of states, $s_I \in S$ is the initial state, X is a *finite* set of *real-valued* parameters, and $\mathcal{P} : S \times S \to \mathbb{Q}[X]$ is a transition function over the states.

The parametric transition probability of going from state s to t is given by $\mathcal{P}(s,t)$. Intuitively, instead of a concrete probability, a transition is equipped with a polynomial over the parameters X. A pMC with $X = \emptyset$ and $\mathcal{P} : S \times S \to \mathit{Distr}(S)$ is a Markov chain (MC). Applying the instantiation $u : X \to \mathbb{R}$ to the pMC $\mathcal{M}$ yields $\mathcal{M}[u]$ by replacing each transition function $f \in \mathbb{Q}[X]$ in $\mathcal{M}$ by $f[u]$. An instantiation u is *well-defined* (for $\mathcal{M}$) if the transition function induced by u yields probability distributions, i.e., if $\mathcal{P}(s,.)[u] \in \mathit{Distr}(S)$, for each $s \in S$. For the well-defined instantiation u, $\mathcal{M}[u]$ is an MC. We assume in the rest of the paper that all instantiations are well-defined. In general, a pMC defines an uncountably infinite family of MCs, where each family member is obtained by a parameter instantiation. (In a similar way, a pBN defines an uncountably infinite set of BNs.)

Example 3. Figure 4(b) depicts an acyclic MC with nine states and initial state *init*. It models our running pregnancy example: initially, there is a possibility of being pregnant (with probability 0.87), or not (with probability 0.13); then, a urine test and a blood test are carried out. Figure 4(a) indicates the parametric version where the *false-negative* probabilities for the *urine test* and the *blood test* are unknown and modeled as parameters

p and q, respectively. The parameter instantiation u with $u(p) = 0.36$, and $u(q) = 0.27$ yields the MC in Fig. 4(b). Let e.g., $\{u : X \to \mathbb{R} \,|\, 0 \leq u(p),\, u(q) \leq 0.1\}$ denote intervals of probability values for the parameters p and q that are related to relatively-precise tests. Such intervals for parameters values are referred to as *regions* later on.

2.3 Symbolic Probabilistic Data Structures

Models such as BNs and MCs can be compactly represented by symbolic data structures such as decision diagrams. We consider two such data structures: *multi-terminal binary decision diagrams* (MTBDDs) and *probabilistic sentential decision diagrams* (PSDDs). We consider *Binary decision diagrams* as the reference to define MTBDDs. The concepts of *vtrees*, *strongly deterministic decomposition*, and *sentential decision diagrams* (SDDs) are introduced as preliminaries to defining PSDD. This section is relevant only to support the experiments in Section 6.2. The definitions are taken from the literature and are only tailored to our notations. The readers who are familiar with MTBDDs and/or PSDDs can skip reading the detailed formal definitions.

Reduced Ordered Binary Decision Diagrams (ROBDDs). ROBDDs, or simply BDDs, represent boolean functions. They result from compacting binary decision trees by eliminating don't care nodes and isomorphic subtrees. The essential characteristic of BDDs is that they are canonical for a given boolean function and a given variable ordering (Bryant, 2018). The size of the BDD strongly depends on the variable ordering. Optimal variable orderings can yield very succinct BDDs. Although finding the optimal variable ordering is NP-hard (Bollig & Wegener, 1996) and mostly is not achieved, BDDs can be very compact in practice (Chaki & Gurfinkel, 2018).

Syntax. Let $\wp = (z_1, \ldots, z_m)$ be a (total) variable ordering for $\mathit{Var} = \{z_1, \ldots, z_m\}$. We also write $z_1 <_\wp \ldots <_\wp z_m$. The *$\wp$-BDD* $\mathfrak{B}$ consists of a finite set V of nodes, partitioned into inner V_I and terminal nodes V_T, and root node $v_0 \in V_I$. The successor functions $\mathit{succ}_0, \mathit{succ}_1 : V_I \to V$ assign a zero-successor and a one-successor to each inner node. The labelling functions $\mathit{var} : V_I \to \mathit{Var}$ and $\mathit{val} : V_T \to \{0, 1\}$ satisfy the following constraint for $v \in V_I$ and $w \in \{\mathit{succ}_0(v), \mathit{succ}_1(v)\}$:

$$(\mathit{var}(v) = z_i \wedge w \in V_I) \;\Rightarrow\; \mathit{var}(w) = z_j \quad \mathit{with} \quad z_i <_\wp z_j.$$

Every inner node v represents a variable z_i and terminal nodes represent 0 or 1.

Semantics. The semantics of *$\wp$-BDD* $\mathfrak{B}$ is the boolean function $f_\mathfrak{B}$ where $f_\mathfrak{B}([z_1 = b_1, \ldots, z_m = b_m])$ is determined by the value of the resulting leaf obtained by traversing the BDD starting from the root v_0 and branching according to the valuation $[z_1 = b_1, \ldots, z_m = b_m]$. For terminal v, f_v represents the constant function f_v with value $\mathit{val}(v)$. For inner node v, f_v is defined by the Shannon expansion $f_v = (\neg z \wedge f_{\mathit{succ}_0}(v)) \vee (z \wedge f_{\mathit{succ}_1}(v))$, where $z{=}\mathit{var}(v)$. Thus, if $b{=}0$, v's next node with $\mathit{var}(v) = z$ is $\mathit{succ}_0(v)$; otherwise $\mathit{succ}_1(v)$. A *$\wp$-BDD* $\mathfrak{B}$ is *reduced* if for every pair (v, w) of nodes in $\mathfrak{B}$, $v \neq w$ implies $f_v \neq f_w$. A BDD can be reduced by recursively applying two reduction rules: eliminating don't care vertices and eliminating isomorphic subtrees.

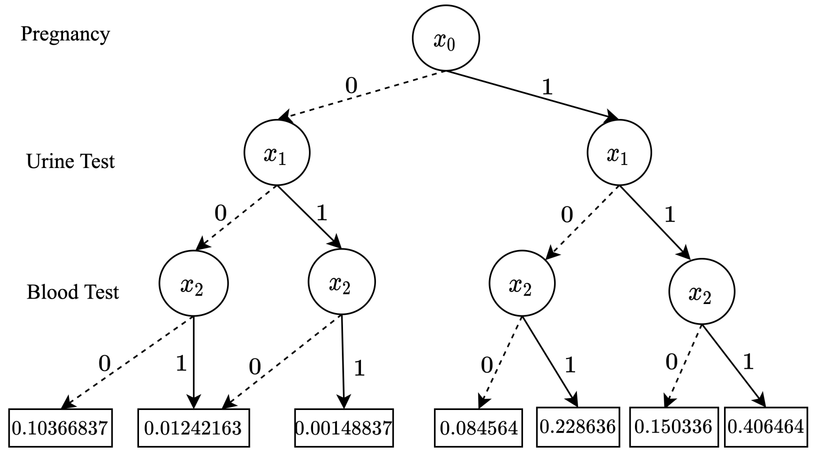


Figure 5: MTBDD of pregnancy example for the variable ordering $P < U < B$.

Multi-Terminal Binary Decision Diagrams. While BDDs represent boolean functions, MTBDDs (Fujita et al., 1997) represent real-valued functions —also known as pseudo-boolean functions. Their terminal values are thus real (or rational) numbers. They are also known as algebraic decision diagrams (Bahar et al., 1997; Sanner & McAllester, 2005). Their semantics is defined analogously to that of BDDs. Let *Var* and $\wp$ be as before. An MTBDD $\mathfrak{M}$ has the same structure as a BDD except that (in our setting) the value function *val* is defined as $\mathit{val}\colon V_T \to [0, 1]$ assigning each terminal node v a probability $\mathit{val}(v)$.

Example 4. Figure 5 shows the MTBDD for the MC in Fig. 4(b) with variable ordering $x_0 < x_1 < x_2$, where the variables correspond to the random variables *Pregnancy*, *Urine Test*, and *Blood Test*. The edges represent the valuation of the variables x_0 through x_2 to 0 (dashed) and 1 (solid). The leaves store the probabilities of the joint pdf. Consider the leftmost path in the MTBDD: $x_0 \xrightarrow{0} x_1 \xrightarrow{0} x_2 \xrightarrow{0} 0.103\ldots$. The value—which is obtained by traversing from the root based on the valuation of each variable—denotes the joint probability of not being pregnant while both the urine and the blood test are negative: $0.10366837 = 0.13 \cdot 0.893 \cdot 0.893$. The second terminal node is shared.

Sentential Decision Diagrams (Darwiche, 2011) represent propositional knowledge bases. They are inspired by two concepts: *strongly deterministic decomposition*: a generalisation of Shannon decomposition (Pipatsrisawat & Darwiche, 2010), and *structured decomposability* (Pipatsrisawat & Darwiche, 2008) that is based on *vtrees*.

A *vtree* (Pipatsrisawat & Darwiche, 2008) for variables *Var* is a full (but not necessarily complete) binary tree whose leaves are in *one-to-one correspondence* with the variables in *Var*. For the vtree node t, let t^l and t^r denote the left and the right child of t, respectively. Let $\mathit{var}(t)$ denote the variables in the leaves of the subtree rooted at node t [1]. The boolean function[2] f *respects* vtree T if for every term $\alpha \wedge \beta$ in f, $\mathit{var}(\alpha) \subseteq \mathit{var}(t^l)$ and $\mathit{var}(\beta) \subseteq \mathit{var}(t^r)$ for some node t in T.

1. For the vtree node t, the subtree rooted at node t is also often denoted by t.
2. in Decomposable Negation Normal Form.

Example 5. Figure 6(a) depicts a vtree for the random variables P, U, and B from the pregnancy example. We have $\mathit{var}(2) = \{P, U, B\}$, $\mathit{var}(4) = \{U, B\}$, and $\mathit{var}(3) = \{U\}$. For instance, the boolean function $f = (U \wedge \neg B)$ respects this vtree.

Strongly deterministic decomposition generalizes Shannon decomposition. Let f be a boolean function over variables Z. For variable $z \in Z$, let $f|z$ and $f|\neg z$ denote the subfunctions that are obtained by setting variable z to true or false in f, respectively. A function f *essentially depends* on variable z iff $f|z \neq f|\neg z$. We write $f(Z)$ to denote that function f only essentially depends on the variables in Z. Let X and Y be a partition of Z. If $f = (p_1(X) \wedge s_1(Y)) \vee \ldots \vee (p_n(X) \wedge s_n(Y))$, then $\{(p_1, s_1), \ldots, (p_n, s_n)\}$ is an (X, Y)-*decomposition* of f in terms of boolean functions p_i and s_i on X and Y. The functions p_i are called primes and the functions s_i are called subs. An (X, Y)-decomposition of function f is *strongly deterministic* on X if $p_i \wedge p_j = \mathit{false}$ for all $i \neq j$. If, in addition, the primes form a partition[3] of X, then it is called an $X-$partition of f. Whereas BDDs are based on the Shannon decomposition of a function f as a $x-$partitioning of f for the single variable x, SDDs are based on $X-$partition of f for some set X of variables.

The SDD node v is *associated with (normalized* [4] *for)* (Darwiche, 2011) the vtree node t according to the following rules :

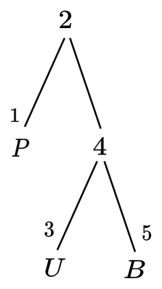
- If v is a terminal node, then t is a leaf node which contains $\mathit{var}(v)$ (if any).
- If v is a decision node, then its primes (subs) are normalized for the left (right) child of t, i.e., t^l and t^r.
- If v is the root SDD node, then t is the root vtree node.

Semantics. Let v be an SDD node that is *associated with* vtree t. The semantics of v is defined by the boolean function f_v as follows.

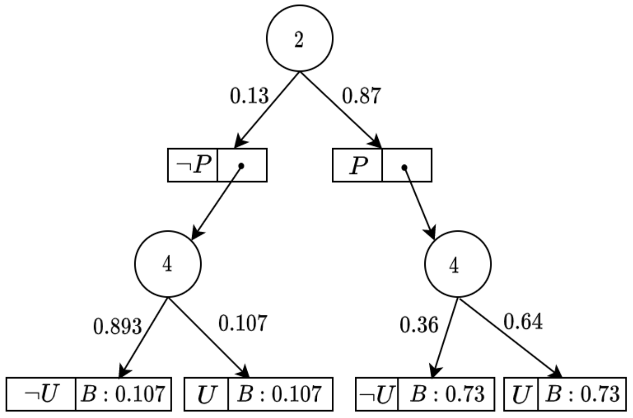
- If v is a terminal node and t is its associated leaf node with $\mathit{var}(t) = z$, then
  - for $v = \bot$, $f_v = \mathit{false}$ and for $v = \top$, $f_v = \mathit{true}$, and
  - for $v = z$, $f_v = z$ and for $v = \neg z$, $f_v = \neg z$.
- If $v = \{(p_1, s_1), \ldots, (p_n, s_n)\}$ is a root or an internal decision node, t is its associated vtree node, $p_1, \ldots, p_n$ are SDDs that are associated with t^l, and $s_1, \ldots, s_n$ are SDDs that are associated with t^r, where $f_{p_1}, \ldots, f_{p_n}$ partition $\mathit{var}(t^l)$. The semantics of v is then given by $f_v = \bigvee_{i=1}^{n} (f_{p_i} \wedge f_{s_i})$.

SDDs are thus a recursive X-partitioning of a boolean function with respect to a given vtree. An SDD is *compressed* if all its subs are distinct. It is *trimmed* if it does not contain any decomposition of the form $\{(\top, \mathfrak{S})\}$ or $\{(\mathfrak{S}, \top), (\neg\mathfrak{S}, \bot)\}$. SDDs that are compressed and trimmed are canonical given a vtree (Darwiche, 2011).

3. Each prime is consistent (i.e., is satisfiable by some evaluation), every pair of distinct primes is mutually exclusive, and the disjunction of all primes holds.
4. As PSDDs are defined based on *normalized* SDDs, we focus the definition on normalized SDDs.



(a) vtree.



(b) PSDD.

Figure 6: A (a) vtree and (b) the PSDD respecting it for the pregnancy example.

Probabilistic Sentential Decision Diagrams. PSDDs (Kisa et al., 2014) extend (normalized) SDDs to represent probability distributions. They are *complete* in the sense that every distribution with finite support can be induced by a PSDD. They are *canonical* as inherited from SDDs: for a given probability distribution and a given vtree, there is a unique trimmed and compressed PSDD. Computing the probability of a term can be done in time linear in the PSDD size (Kisa et al., 2014).

Syntax. A PSDD decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ is an SDD decision node (p_i, s_i), $\ldots, (p_k, s_k)$, where every prime p_i is equipped with a probability θ_i. We have $\theta_1 + \ldots + \theta_k = 1$ and $\theta_i = 0$ iff $s_i = \bot$. Each terminal node $\top$ is equipped with probability θ where $0 < \theta < 1$. For the internal decision nodes, probability θ_i labels the incoming edge to (p_i, s_i). For the terminal nodes, it is denoted by $z\colon \theta$, where z is the variable of the vtree leaf node that $\top$ is normalized for. Other terminal nodes ($\bot$, z, and $\neg z$) have fixed pre-defined probabilities.

Semantics. Let v be a PSDD node that is *associated with* vtree node t. The semantics of v is defined as the probability distribution $\Pr_v$ over the variables in $\mathit{var}(t)$ as follows.

- If v is a terminal node and $\mathit{var}(t) = z$, then
  - for $v = z\colon \theta$, $\Pr_v(z) := \theta$ and $\Pr_v(\neg z) := 1 - \theta$,
  - for $v = \bot$, $\Pr_v(z) := 0$ and $\Pr_v(\neg z) := 0$,
- If v is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ that is normalized for the vtree node t with $\mathit{var}(t^l) = X$ and $\mathit{var}(t^r) = Y$, then $\Pr_v(\overline{X}, \overline{Y}) := \Pr_{p_i}(\overline{X}) \cdot \Pr_{s_i}(\overline{Y}) \cdot \theta_i$, for i with $\overline{X} \models p_i$. Here, $\overline{X} \in \mathit{Eval}(X)$ is a joint evaluation for variables in X. Note that, by the definition of strongly deterministic decomposition, the condition $\overline{X} \models p_i$ holds on exactly one of the primes: primes are mutually exclusive.

Example 6. Figure 6 (b) indicates the PSDD for the pregnancy example (from Fig. 2) that is associated with the vtree in Fig. 6 (a). Circles denote decision nodes and rectangles denote prime-sub pairs (p_i, s_i). The decision nodes are linked to the prime-sub pairs by the edges that are labeled with probability values (θ_i). The inner node 4 is, e.g., an (X, Y)-decomposition for $X=\{U\}$ and $Y=\{B\}$. By PSDD semantics, we obtain $\Pr(P = no \wedge U = neg \wedge B = neg) = 0.13 \cdot 0.893 \cdot (1-0.107) = 0.10366837$.

3. Objectives for Parametric Models

Whereas the previous section introduced the probabilistic models relevant to this paper, we now present some objectives on (parametric) Bayesian networks and parametric Markov chains. These objectives indicate what the synthesis problems are that we focus on. The typical perspective is that a requirement is given and that we are interested in finding some (or all) parameter values that fulfill this requirement, or that no parameter value can fulfill the requirement.

3.1 Objectives for Parametric Bayesian Networks

We discuss four synthesis problems for pBNs by considering their corresponding decision problems (Kwisthout & Van der Gaag, 2008). We took the objectives explicitly from the BN literature (Castillo et al., 1997a; Coupé & Van der Gaag, 2002; Kwisthout & Van der Gaag, 2008) and tailored them to our notation. In the sequel, let $\Pr_{\mathcal{B}}$ denote the parametric joint distribution function induced by pBN $\mathcal{B} = (V, E, X, \Theta)$ and $\Pr_{\mathcal{B}}[u]$ the joint probability distribution of BN $\mathcal{B}[u]$ at well-defined instantiation $u : X \to \mathbb{R}$. Let $E \subseteq V$ be the *evidence*, $H \subseteq V$ the *hypothesis*, and $\lambda \in \mathbb{Q} \cap [0, 1]$ a rational *threshold*. Let $e, e' \in \mathit{Eval}(E)$ and $h, h' \in \mathit{Eval}(H)$ be some joint evaluations for the variable in E and H. We start off with a well-known objective for classical BNs.

Threshold inference. Threshold inference for classical BN $\mathcal{B}$ amounts to check whether $\Pr_{\mathcal{B}}(H = h \mid E = e) \sim \lambda$ for threshold λ and comparison operator $\sim \in \{\leq, \geq\}$. A simplified version is obtained by only considering an evidence, i.e., whether $\Pr_{\mathcal{B}}(E = e) \sim \lambda$. The objective is concerned with determining the function that describes $\Pr_{\mathcal{B}}$.

Sensitivity function. For the evidence $E{=}e$ and hypothesis $H{=}h$[5], determine the sensitivity function $f_{\Pr_{\mathcal{B}}(H=h|E=e)}$, such that for every instantiation $u : X \to \mathbb{R}$

$$f_{\Pr_{\mathcal{B}}(H=h|E=e)}[u] \;=\; \Pr_{\mathcal{B}[u]}(H = h \mid E = e).$$

Due to the presence of parameters, the function $f_{\Pr_{\mathcal{B}}}$ is a rational function over X, i.e., a fraction g/h with $g, h \in \mathbb{Q}[X]$. This function reduces to a probability distribution for well-defined instantiation u. In a similar way, the function $f_{\Pr_{\mathcal{B}}(E=e)}$ is defined.

Example 7. Consider the pBN $\mathcal{B}$ for our running example (Fig. 3) and assume that we are interested in the probability of a cow being pregnant given that both tests are negative. The sensitivity function for this objective is:

$$f_{\Pr_{\mathcal{B}}(\mathit{Pregnancy}=\mathit{yes}|\,\mathit{UrineTest}=\mathit{neg}\wedge\mathit{BloodTest}=\mathit{neg})} \;=\; \frac{87000000{\cdot}p{\cdot}q}{87000000{\cdot}p{\cdot}q + 10366837}. \qquad (1)$$

5. As E and H are sets of vertices, both e and h are in fact vectors of values.

Parameter tuning. For pBN $\mathcal{B}$, a natural objective is to *find an instantiation* u with $\Pr_{\mathcal{B}}[u](H = h \mid E = e) \sim \lambda$ or report that such an instantiation does not exist.

Example 8. Consider again the pBN of our running example (Fig. 3) and assume that the farmer requires that the probability of a cow being pregnant given that both tests are negative to be below 1/5. By equation (1), this yields $p \cdot q \leq 0.027$.

The next two objectives capture the change if the hypothesis $H{=}h$ is adapted to $H{=}h'$.

Hypothesis ratio parameter tuning. Given the hypotheses $H{=}h$, $H{=}h'$, and $\lambda \in \mathbb{Q}^+$ for evidence $E{=}e$, find an instantiation u such that:

$$\frac{\Pr_{\mathcal{B}}[u](H = h' \mid E = e)}{\Pr_{\mathcal{B}}[u](H = h \mid E = e)} \sim \lambda \quad \text{i.e.,} \quad \frac{\Pr_{\mathcal{B}}[u](H = h' \wedge E = e)}{\Pr_{\mathcal{B}}[u](H = h \wedge E = e)} \sim \lambda. \tag{2}$$

Example 9. For the pBN of our running example (Fig. 3), assume that the farmer intend to ensure the following constraint:

$$\frac{\Pr_{\mathcal{B}}(\mathit{UrineTest{=}pos} \wedge \mathit{BloodTest{=}pos} \mid \mathit{Pregnancy{=}yes})}{\Pr_{\mathcal{B}}(\mathit{UrineTest{=}neg} \wedge \mathit{BloodTest{=}neg} \mid \mathit{Pregnancy{=}yes})} \geq 9,$$

which can be simplified to the following equation:

$$\frac{\Pr_{\mathcal{B}}(\mathit{UrineTest{=}pos} \wedge \mathit{BloodTest{=}pos} \wedge \mathit{Pregnancy{=}yes})}{\Pr_{\mathcal{B}}(\mathit{UrineTest{=}neg} \wedge \mathit{BloodTest{=}neg} \wedge \mathit{Pregnancy{=}yes})} \geq 9.$$

This yields $1 - p - q - 8 \cdot p \cdot q \geq 0$. The equation does not hold for the the original values of p and q in the original BN (Fig. 2). Tuning the parameters to the parameter instantiation u with $u(p) = 0.3$ and $u(q) = 0.2$ satisfies the constraint.

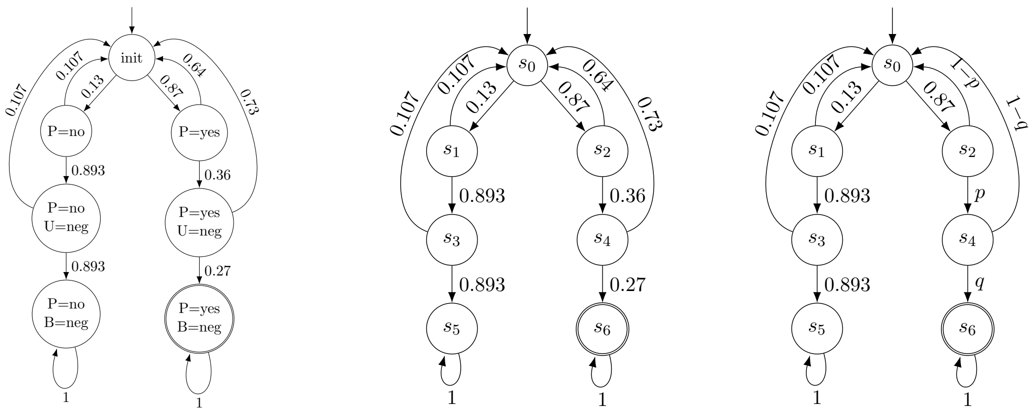
Hypothesis difference parameter tuning. Given the hypotheses $H{=}h$ and $H{=}h'$ for evidence $E{=}e$, find an instantiation u such that:

$$\Pr_{\mathcal{B}}[u](H = h \mid E = e) \; - \; \Pr_{\mathcal{B}}[u](H = h' \mid E = e) \; \sim \; \lambda.$$

The difference and ratio problems can analogously be defined for varying evidences rather than varying hypotheses. The *evidence* tuning problems are defined for values e and e' for E, given a fixed value h for H.

3.2 Objectives on Parametric Markov Chains

Let $\mathcal{D} = (S, s_I, X, \mathcal{P})$ with $X = \emptyset$ be a classical non-parametric MC. Let $\mathit{Paths}(s)$ denote the set of all infinite paths in $\mathcal{D}$ starting from s, i.e., all infinite sequences of the form $s_1 s_2 s_3 \ldots$ with $s_1 = s$ and $\mathcal{P}(s_i, s_{i+1}) > 0$. A probability measure $\Pr_{\mathcal{D}}$ is defined on measurable sets of infinite paths using a standard cylinder construction; for details, see, e.g., (Baier & Katoen, 2008, Ch. 10).



(a) MC. (b) MC - abstracted. (c) pMC - abstracted.

Figure 7: The MC (a,b) and pMC (c) for pregnancy example tailored to the evidence Urine Test = neg and Blood Test = neg. (b) and (c) are abstracted from variable-valuations.

Reachability probabilities. For $\mathcal{D}$ and $s \in S$, the *reachability probability* of the set $G \subseteq S$ of goal (target) states, denoted using standard notation from temporal logic by $\Pr_{\mathcal{D}}(\Diamond G)$, is defined as the probability to eventually reach some state in G from s:

$$\Pr_{\mathcal{D}}(\Diamond G) \;=\; \Pr_{\mathcal{D}}\{\, s_1 s_2 s_3 \ldots \in \mathit{Paths}(s) \mid \exists i.\, s_i \in G \,\}. \tag{3}$$

We are often interested in the reachability probability from the initial state s_I in $\mathcal{D}$. From now on, $s = s_I$ when not explicitly mentioned otherwise. Reachability probabilities can be obtained as the unique solution of a linear equation system (Baier & Katoen, 2008) whose size is proportional to the number of states in $\mathcal{D}$.

Example 10. Consider the MC $\mathcal{D}$ in Fig. 7(b). We refer to Sec. 4 on how the MC is obtained. Let p_{s_i} denote the probability of reaching target state s_6 from state s_i and assume we are interested in eventually reaching s_6 from the initial state s_0 with a probability below the threshold $\lambda = 1/5$, i.e., $\Pr(\Diamond s_6) \leq 1/5$. We refer to this as the specification φ. The reachability probability $\Pr(\Diamond s_6)$ is obtained by solving the linear equation system:

$$\begin{aligned} p_{s_0} &= 0.13 \cdot p_{s_1} + 0.87 \cdot p_{s_2} & p_{s_3} &= 0.893 \cdot p_{s_5} + 0.107 \cdot p_{s_0} \\ p_{s_1} &= 0.893 \cdot p_{s_3} + 0.107 \cdot p_{s_0} & p_{s_4} &= 0.27 \cdot p_{s_6} + 0.73 \cdot p_{s_0} \\ p_{s_2} &= 0.36 \cdot p_{s_4} + 0.64 \cdot p_{s_0} & p_{s_5} &= 0, \text{ and } p_{s_6} = 1. \end{aligned}$$

Verifying the specification φ is equivalent to checking whether $p_{s_0} \leq 1/5$. Solving the equation system yields $p_{s_0} = 0.4925323$. Thus MC $\mathcal{D}$ refutes φ.

Solution functions. The solution function $\Pr_{\mathcal{M}}(\Diamond G)$ for pMC $\mathcal{M}$ is a rational function over the parameters in X such that for every instantiation u it holds that (Daws, 2004):

$$\Pr_{\mathcal{M}}(\Diamond G)[u] = \Pr_{\mathcal{D}}(\Diamond G) \text{ where MC } \mathcal{D} = \mathcal{M}[u].$$

Example 11. Consider the pMC $\mathcal{M}$ in Fig. 7(c). The rational function $\Pr(\Diamond s_6)$ is obtained by solving the *non-linear* equation system that differs from the above linear equation only for the two parametric states s_2 and s_4:

$$p_{s_2} = p \cdot p_{s_4} + (1{-}p) \cdot p_{s_0} \qquad\qquad p_{s_4} = q \cdot p_{s_6} + (1{-}q) \cdot p_{s_0}$$

The unique solution for state s_0 of this equation system is the required solution function:

$$\Pr(\Diamond s_6) \;=\; p_{s_0} \;=\; \frac{87000000{\cdot}p{\cdot}q}{87000000{\cdot}p{\cdot}q + 10366837}. \tag{4}$$

Notice that this is exactly the function obtained in Example 7. We will see later that such correspondence holds in general.

Let pMC $\mathcal{M} = (S, s_I, X, \mathcal{P})$ with goal states $G \subseteq S$, $\lambda \in \mathbb{Q} \cap [0,1]$ a threshold, and $\sim \in \{\leq, \geq\}$ be a comparison operator. Assume the specification objective is to reach a state in $G \subseteq S$ with a probability that meets the constraint $\sim \lambda$. Formally, the fact that pMC $\mathcal{M}$ satisfies the specification φ for the parameter instantiation $u : X \to \mathbb{R}$, denoted $\mathcal{M}[u] \models \varphi$, is defined by:

$$\mathcal{M}[u] \models \varphi \quad \text{if and only if} \quad \Pr_{\mathcal{M}[u]}(\Diamond G) \sim \lambda.$$

Mostly, we are not interested in one specific instantiation u but rather in an entire set of such instantiations—such set is called a *region*. Let $R \subseteq \mathbb{R}^n$ be a region, where n is the number of parameters in $\mathcal{M}$, i.e., $n = |X|$ and let

$$\mathcal{M}, R \models \varphi \quad \text{if and only if} \quad \forall u \in R.\ \mathcal{M}[u] \models \varphi.$$

Feasibility problem. Find an instantiation u such that $\mathcal{M}[u] \models \varphi$ or report that such an instantiation does not exist.[6]

Example 12. For the example pMC in Fig. 7(c) and the constraint $\Pr(\Diamond s_6) \leq 1/5$, the feasibility problem is equivalent to finding a solution to the formula:

$\exists\, p_{s_0}, \ldots, p_{s_6}, p, q \in \mathbb{R}$ such that

$$\begin{array}{ll}
p_{s_0} \leq 1/5 & \wedge\ p_{s_2} = p \cdot p_{s_4} + (1{-}p) \cdot p_{s_0} \\
\wedge\ 0 < p < 1\ \wedge\ 0 < q < 1 & \wedge\ p_{s_3} = 0.893 \cdot p_{s_5} + 0.107 \cdot p_{s_0} \\
\wedge\ p_{s_0} = 0.13 \cdot p_{s_1} + 0.87 \cdot p_{s_2} & \wedge\ p_{s_4} = q \cdot p_{s_6} + (1{-}q) \cdot p_{s_0} \\
\wedge\ p_{s_1} = 0.893 \cdot p_{s_3} + 0.107 \cdot p_{s_0} & \wedge\ p_{s_5} = 0, \wedge p_{s_6} = 1.
\end{array}$$

This is a formula in the existential theory of reals (ETR for short)[7]. The above ETR-formula reduces to:

$$\exists p, q \in \mathbb{R}.\ 0 < p, q < 1\ \wedge\ \frac{87000000{\cdot}p{\cdot}q}{87000000{\cdot}p{\cdot}q + 10366837} \;\leq\; 1/5. \tag{5}$$

6. The *optimal* feasibility problem is to find an instantiation u that maximises (or dually minimises) the probability to satisfy φ.
7. Each feasibility problem on pMCs can be encoded as ETR-formula. In addition, each ETR-formula ψ can be encoded as a reachability constraint φ on an pMC $\mathcal{M}$ (that depends on ψ) such that ψ has a solution if and only if $\mathcal{M} \models \varphi$. The feasibility problem for pMCs is thus *ETR-complete* (Junges, Katoen, Pérez, & Winkler, 2021). ETR is a complexity class that lies in between NP and PSPACE.

Parameter space partitioning. The aim is to partition region R into R_+, R_-, and $R_?$ such that:

$$R_+ \subseteq \underbrace{\{u \in R \mid \mathcal{M}[u] \models \varphi\}}_{\text{satisfying instantiations}} \quad \text{and} \quad R_- \subseteq \underbrace{\{u \in R \mid \mathcal{M}[u] \models \neg\varphi\}}_{\text{refuting instantiations}}$$

and $R_? = R \setminus (R_+ \cup R_-)$ with $||R_?|| \leq (1-\eta)\cdot||R||$ for some given *coverage factor* $0 \leq \eta \leq 1$. The sub-region $R_?$ denotes the fragment of R that is *inconclusive* for φ. It is required that this fragment covers at most a factor $1-\eta$ of the size of R. An *exact* parameter space partitioning is obtained when η=1, i.e., if:

$$R_+ = \{u \in R \mid \mathcal{M}[u] \models \varphi\}, \quad R_- = \{u \in R \mid \mathcal{M}[u] \models \neg\varphi\}, \quad \text{and} \quad R_? = \emptyset.$$

Thus, then all instantiations satisfying (and refuting) φ are determined.

A problem naturally coming up for partitioning is the *region verification problem*, which for $R_{sub} \subseteq R$ is to check whether:

$$\underbrace{\mathcal{M}, R_{sub} \models \varphi}_{R_{sub} \text{ is accepting}} \quad \text{or} \quad \underbrace{\mathcal{M}, R_{sub} \models \neg\varphi}_{R_{sub} \text{ is rejecting}} \quad \text{or} \quad \underbrace{\mathcal{M}, R_{sub} \not\models \varphi \wedge \mathcal{M}, R_{sub} \not\models \neg\varphi}_{R_{sub} \text{ is inconclusive}}.$$

Example 13. Consider our running example from Fig. 7(c) with the solution function $f(p,q) = \Pr(\Diamond s_6)$ as obtained in equation (4). Let $0 < p, q < 1$ and R be the two-dimensional hyper-rectangle $[0,1] \times [0,1]$. For the specification $\Pr(\Diamond s_6) \leq 1/5$, we have:

$$R_+ = \{u \mid f[u] \leq 1/5\} \quad \text{and} \quad R_- = R \setminus R_+ = \{u \mid f[u] > 1/5\}.$$

In terms of p and q, R_+ coincides with the solution space of $p \cdot q \leq 0.027$; see Example 8.

4. From Bayesian Networks to Markov Chains

The key of our approach to tackle various synthesis and inference problems on (p)BNs is to exploit model-checking techniques on MCs (Baier & Katoen, 2008; Katoen, 2016; Baier et al., 2018) and synthesis techniques (Junges et al., 2019) on pMCs. To that end, we transform a (p)BN into a (p)MC. We first introduce a transformation that is applicable to all objectives on pBNs. This *evidence-agnostic* transformation models the possible valuations of the random variables in the BN as MC states. We then propose a second transformation that is tailored to the evidence in an inference query. The latter *evidence-tailored* transformation is inspired by (Baier, Klein, Klüppelholz, & Märcker, 2014). We intuitively explain the transformations through an example and and refer the reader to the Appendix A for further details and proofs.

Evidence-agnostic transformation. This mapping takes the (p)BN $\mathcal{B} = (V, W, X, \Theta)$ and a topological ordering ϱ on the DAG (V, W), and constructs the (p)MC $\mathcal{M}_{\mathcal{B}}^{\varrho} = (S, s_0, X, \mathcal{P})$ over the same set of parameters X. The transition probability functions in the pMC are in correspondence with the CPT entries, while the states correspond to the possible valuations of the vertices in the pBN. We illustrate this through an example.

Example 14. Figure 4(a) indicates the pMC for the pBN of our running example for the topological ordering $\varrho = P <_\varrho U <_\varrho B$. For simplicity, the variables are denoted by their first letters and the *don't care* evaluations of variables are omitted. The initial state represents that all variables are *don't care.* It has two outgoing transitions to the states *Pregnancy* = *yes* and *Pregnancy* = *no* with probabilities 0.87 and 0.13 as given by the CPT entries of vertex *Pregnancy* in the pBN, the first variable in the topological order ϱ. In general, at "level" i of the MC, the outgoing transitions are determined by the CPT of the i+1-st variable in the topological order.

The transformation possesses the property that inference objectives on (p)BN $\mathcal{B}$ can be reduced to corresponding reachability queries on the (p)MC $\mathcal{M}_{\mathcal{B}}^{\varrho}$. Let E and H be logical formulas over the propositions $(v = d)$ with $v \in V$ and $d \in D_v$.

Proposition 1. The evidence-agnostic pMC $\mathcal{M}_{\mathcal{B}}^{\varrho}$ of pBN $\mathcal{B}$ satisfies:

$$\Pr_{\mathcal{B}}(E) \;=\; 1 - \Pr_{\mathcal{M}_{\mathcal{B}}^{\varrho}}(\lozenge\, \neg E) \quad \text{and} \quad \Pr_{\mathcal{B}}(H \mid E) \;=\; \frac{1 - \Pr_{\mathcal{M}_{\mathcal{B}}^{\varrho}}(\lozenge\,(\neg H \vee \neg E))}{1 - \Pr_{\mathcal{M}_{\mathcal{B}}^{\varrho}}(\lozenge\, \neg E)},$$

where the latter equality requires $\Pr_{\mathcal{M}_{\mathcal{B}}^{\varrho}}(\lozenge\, \neg E) < 1$. See the proof in Appendix A.

Thus, exact inference on pBNs is reduced to computing reachability probabilities on the corresponding pMC.

Evidence-tailored transformation. This mapping aims at obtaining more compact pMCs by exploiting the evidence at hand. Let pBN $\mathcal{B}$ and ϱ be a topological order on its set V of vertices. Let $E = (v_{E_1}{=}d_{E_1}) \wedge \cdots \wedge (v_{E_k}{=}d_{E_k})$ be the evidence with $v_{E_1} <_\varrho \ldots <_\varrho v_{E_k}$ and $\mathit{vars}(E) = \{v_{E_1}, \ldots, v_{E_k}\}$. In the sequel, we denote E_k as E_{last}. The E-tailored pMC $\mathcal{M}_{\mathcal{B},E}^{\varrho}$ is obtained by the following two amendments to the evidence-agnostic pMC $\mathcal{M}_{\mathcal{B}}^{\varrho}$:

1. *Propagation operation:* Let $v_j \in V$ with $v_j \notin \mathit{vars}(E)$ and $v_j <_\varrho v_{E_{last}}$. We propagate the value of v_j until the level $\varrho(v_{E_{last}})$.
2. *Redirection operation :* Let $S_{\neg E}$ denote the states in $\mathcal{M}_{\mathcal{B}}$ that violate the evidence E. We delete the incoming transitions to every state in $S_{\neg E}$ and redirect them to the initial state.

Example 15. Figure 7(a) on page 17 is the E-tailored MC obtained for the pregnancy BN and the evidence E = *U=neg* $\wedge$ *B=neg.* The MC reduces the MC in Fig. 4(b) by (1) propagating the values of P up to the last evidence level and (2) deleting the edges leading to *U=pos* or *B=pos* and redirecting them to the initial state. Unreachable states are subsequently deleted.

Intuitively speaking, the transitions towards the *evidence-violating* states are in the evidence-tailored pMC redirected to the initial state.

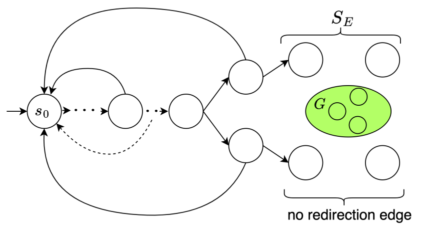


Figure 8: Evidence-tailored general sketch.

Let H be a logical formula over the atomic propositions $(v_{H_i} = d_{H_i})$ that is decomposable to the sub-formulas H_{before} and H_{after}, such that the variables appearing in H_{before} all occur before $v_{E_{last}}$ in the topological order ϱ and the variables involved in H_{after} all occur after $v_{E_{last}}$. Without loss of generality, let $H = (v_{H_1}{=}d_{H_1}) \wedge \cdots \wedge (v_{H_l}{=}d_{H_l})$ be the hypothesis. Let $H_{before} = \bigwedge_{i=1}^{b}(v_{H_i}{=}d_{H_i})$ with $v_{H_i} \leq_\varrho v_{E_{last}}$ and let $H_{after} = \bigwedge_{i=a}^{l}(v_{H_i}{=}d_{H_i})$ with $v_{H_i} >_\varrho v_{E_{last}}$, such that $H = H_{before} \wedge H_{after}$.

Proposition 2. The evidence-tailored pMC $\mathcal{M}^{\varrho}_{\mathcal{B},E}$ of pBN $\mathcal{B}$ ensures that

$$\Pr_{\mathcal{B}}(H \mid E) \ = \ 1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}} \big(\Diamond\big((\neg H_{before} \wedge E_{last}) \vee \neg H_{after}\big)\big).$$

Example 16. Consider the evidence-tailored MC in Fig. 7(a). We have by Proposition 2, $\Pr_{\mathcal{B}}(P{=}yes \,|\, U{=}neg \wedge B{=}neg) = 1 - \Pr_{\mathcal{M}_{\mathcal{B},E}}(\Diamond(P{=}no \wedge B{=}neg)) = 1 - \Pr(\Diamond s_5)$ that amounts to $\Pr(\Diamond s_6)$ in this example with a single hypothesis node.

The key to Proposition 2 is that the goal states occur after the last redirection edge, see Fig. 8. This is ensured by the propagation operation. The redirection operation ensures that for such goal states the obtained probability distribution is normalized by the probability of the evidence. We refer to App. A for the detailed proofs.

Again, exact inference on pBNs is reduced to reachability probabilities in pMC. This result facilitates using the algorithms for pMC analysis for pBN parameter tuning. The next section surveys the main state-of-the-art algorithms for analysing parametric MCs.

5. Probabilistic Model-Checking Algorithms

Probabilistic model checking (PMC) is a fully automated technique to analyse quantitative properties of Markov chains and extensions thereof (with non-determinism) such as Markov decision processes. The properties are usually formalised using probabilistic temporal logic such as Probabilistic Computation Tree Logic (PCTL) (Hansson & Jonsson, 1994). PCTL is a very flexible and expressive language to describe a wide range of properties. The key procedure to verify an MC against a temporal logic formula is to compute reachability probabilities. As illustrated in Example 10 (page 1651), computing reachability probabilities amounts to computing the unique solution of a linear equation system whose size is linear in the number of states in the MC. Thus, reachability probabilities can be computed in polynomial time in the size of the MC. Reachability properties are the central notion used in this paper. Model-checking results include (1) quantitative results such as the probabilities to reach the states of interest (from each state in the MC), (2) qualitative results: whether the threshold is met or not, and (3) a counterexample, i.e., a sub-MC—possibly provided as syntactical description—that refutes the property, in case the threshold is violated (Wimmer, Jansen, Vorpahl, Ábrahám, Katoen, & Becker, 2015).

Like any state-based analysis technique, the state explosion problem—how to treat models with a large number of states?—is a challenge naturally rising in (probabilistic) model checking. Two leading techniques to overcome this challenge are *symbolic model checking* and *bisimulation minimization.* These approaches are originally exploited in classical model checking and have been carried over to the probabilistic and the parametric setting. The key to *symbolic model checking* is to represent the MC as an MTBDD (see Section 2.3) that captures the symmetries and redundancies in the model. The usage of MTBDDs in probabilistic model checking originates from (Baier et al., 1997) and is nowadays supported by the key probabilistic model checking tools such as PRISM (Kwiatkowska et al., 2011) and Storm (Dehnert et al., 2017).

The idea of *bisimulation minimization* is to merge the states that have equal transition probabilities to all the equivalent classes that are defined with respect to an equivalence relation on the state space. Bisimulation preserves the validity of all formulas that can be expressed in PCTL which makes it an attractive minimization technique. In addition, the bisimulation quotient, i.e., the aggregated state space, (a) can be obtained *efficiently* (Valmari & Franceschinis, 2010)[8], and (b) is the *coarsest* partitioning of the state space that preserves the validity of the property of interest (Baier & Katoen, 2008, Ch. 10). That is to say, any coarser partitioning has to abstract from some information such that the validity of the property of interest cannot be ensured anymore. The use of bisimulation minimization leads to a substantial efficiency gain (Katoen, Kemna, Zapreev, & Jansen, 2007).

6. Experimental Results: The Non-Parametric Setting

We have developed a prototypical tool on top of Storm (Dehnert et al., 2017), the probabilistic model checker that dominated the last and only two model-checking competitions, see `qcomp.org`. Our tool implements both the evidence-agnostic and evidence-tailored (p)BN to (p)MC transformations as described in Section 4. The (p)BNs are expressed in the `bif` format and the (p)MCs are either encoded in Jani (Budde et al., 2017) or in the explicit `drn` format. As Jani is an intermediate language, other probabilistic model-checking tools such as `mcsta` (Hartmanns & Hermanns, 2014), EPMC (Fu et al., 2022) and PRISM (Kwiatkowska et al., 2011) can be used as backend verifiers too. We used Storm because of its efficiency, its modularity and its `python` interface enabling fast prototyping while using various of the modules within Storm.

8. The complexity is in $\mathcal{O}(m \cdot \log n)$ where n is the number of states in the MC and m the number of transitions.

network	#nodes	#edges	Dmax	AMB
cancer	5	4	2	2.00
earthquake	5	4	2	2.00
asia	8	8	2	2.5
survey	6	6	3	2.67
child	20	25	6	3.00
sachs	11	17	3	3.09
alarm	37	46	4	3.51
hailfinder	56	66	11	3.54
pathfinder	135	200	63	3.81
hepar2	70	123	4	4.51
mildew	35	46	100	4.57
insurance	27	52	5	5.19
barley	48	84	67	5.25
andes	223	338	2	5.61
win95pts	76	112	2	5.92
water	32	66	4	7.69

Table 1: Statistics on the BN benchmarks taken from `bnlearn`.

We took the BN benchmarks from the `bnlearn` repository (Scutari, 2019) and conducted our experiments on a 2.3 GHz Intel Core i5 processor with 16 GB RAM. We focused our experiments on various research questions. Table 1 overviews the benchmarks we have used for our experiments. The first column refers to the name of the BN, the second and the third indicate the number of BN nodes and the number of BN edges. The fourth column refers to the maximum domain of the BN variables. The last column shows the average Markov blanket; a metric that shows the degree of dependency between the BN variables and is an indicator of the practical complexity of inference in BNs.

This section covers the experimental results for the non-parametric setting, which includes *probabilistic inference*, *symbolic probabilistic inference*, and evaluating the *evidence-tailored translation.*

6.1 Probabilistic Inference

Baseline. To evaluate our PMC-based tool for performing probabilistic inference on BNs, we took the state-of-the-art BN inference tool Ace as baseline. *Ace*[9] is developed by Darwiche's group that is the state-of-the-art for probabilistic inference on Bayesian networks; see the recent study (Agrawal, Pote, & Meel, 2021). It takes a BN as input and compiles it into an arithmetic circuit (AC). The generated AC is used to compute multiple inference queries; the so-called online inference. The circuit represents the BN marginals as polynomials over BN variables and CPT entries. The leaves of the AC store the inference indicators (for each BN variable valuation) as well as the CPT entries; the intermediate nodes store the arithmetic operators: summations and multiplications. This allows to directly and

9. `http://reasoning.cs.ucla.edu/ace/`

	Storm	Ace
computing prior and posterior marginals	✓	✓
computing most probable explanation	✓	✓
supporting evidence-tailored compilation	✓	✓
efficient inference on some very large networks	✗	✓
supporting arbitrary formulas for evidence and hypothesis	✓	✗

Table 2: Capabilities of Storm and Ace.

efficiently compute the probability of the evidence; i.e., $\Pr(E{=}e)$ on the root as well as the marginal probabilities of individual BN variable-valuations along the path from the leaves to the root; i.e., $\Pr(H{=}h \mid E{=}e)$, when H only contains one variable. Being mainly specific to these queries and taking context-specific independence into account makes the AC-based approach quite efficient. Ace however lacks support for posing more general forms of queries on the compiled AC, e.g., it does not support hypotheses involving (disjunctions and conjunctions over) multiple random variables. In our experiments, we therefore consider unary hypotheses and the more general case separately. Its capabilities and restrictions are summarized in Table 2. The first three experiments consider the simple translation from BNs to MCs, i.e., the evidence-agnostic mapping. The fourth experiment focuses on the effects of the evidence-tailored mapping.

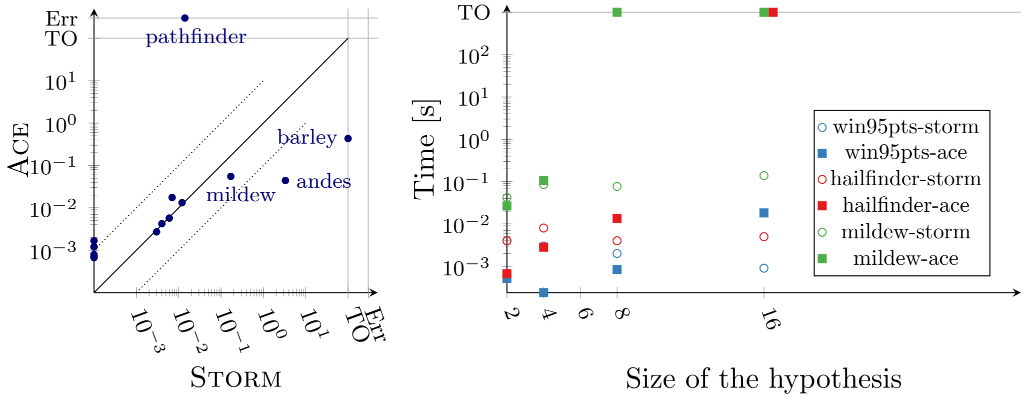
RQ1: *How does PMC-based inference compare to Ace, for classical queries with unary hypothesis?*

We performed a series of experiments to investigate how the performance of BN inference via model checking MCs compares to Ace. The considered benchmarks are taken from the set of *small*, *medium*, *large*, and *very large* networks in the `bnlearn` repository. Figure 9(a) indicates our results in a log-log scale plot. Each point represents the time (in seconds) to compute the marginal probabilities for every valuation of every random variable of the BN at hand. The x-axis indicates the total model-checking time for computing the marginals by Storm, and the y-axis indicates the corresponding total inference time by Ace. The pre-processing and compilation times for both approaches are excluded. Err denotes a run-time error and TO indicates a time-out (of 15 min).

The results indicate that for most of the benchmarks Storm has the same or slightly less inference time as Ace. For the very large network `pathfinder`, Ace threw a run-time error; Storm computed the marginals in 0.014s. For the networks `andes`, `mildew`, and `barley`, Ace outperformed Storm. This is very likely related to the so-called *context-specific* independence in these benchmarks: more than 95% of the total 540,150 CPT entries of `mildew` are repeated probability values and—to a more extreme extent—only 19 distinct probability values occur in totally 1157 CPT entries of `andes`. A similar characteristic can be observed in `barley`: `barley` has a random variable with an exceptionally large domain (67), yet all the CPT entries of this node are equal. Our (p)BN to (p)MC translation is not optimized for inference with multiple occurrences of probability values in the CPTs, while in the AC-based approach the CPT entries are stored in the leaves and are shared.

RQ2: *How does PMC-based inference compare to Ace, for hypotheses involving multiple BN variables?*

We now consider queries of the form $\Pr(H \mid E)$ where H (and E) involve arbitrarily many variables, and conjunctions and/or disjunctions thereof such as e.g., $\Pr((v_1{=}d_1) \vee \cdots \vee (v_k{=}d_k))$ or $\Pr((v_1{=}d_1 \vee v_1{=}d_1') \wedge \cdots \wedge (v_k{=}d_k \vee v_k{=}d_k' \vee v_k{=}d_k''))$. Such queries can be directly translated into corresponding reachability queries for model checking and do not require any special treatment. In order to handle such queries in Ace, one has to extend the BN with an auxiliary variable for the entire hypothesis, e.g., for the formula $(v_1{=}d_1) \vee \cdots \vee (v_k{=}d_k)$, and define its CPT accordingly [10]. Treating such queries thus does come at the expense of enlarging the BN at hand. Figure 9(b) shows the results for hypotheses involving the



(a) unary hypothesis (b) n-ary hypothesis

Figure 9: Storm vs. Ace: probabilistic inference for (a) $|H|{=}1$ and (b) $|H|{=}n$.

disjunction of multiple variables for the benchmarks `win95pts`, `hailfinder`, and `mildew`. We varied the number of variables in the disjunction from one to 16. The x-axis indicates the hypothesis size in terms of the number of BN nodes. The y-axis (in log-scale) indicates the inference time in seconds. TO indicates a time-out of 15 minutes. The pre-processing time for adding the auxiliary variable and recompiling the network for Ace is not incorporated. Storm's performance is not influenced by increasing the hypothesis size. On the contrary, the inference time for Ace increases as the BN size increases when enlarging the hypothesis. Adding a new auxiliary node to the BN can be quite expensive: the auxiliary node's CPT is exponential in the number of hypothesis nodes, and also the tree-width of the BN increases. These lead to an adverse effect on the inference time for Ace.

6.2 Symbolic Probabilistic Inference

For the first two research questions, we used Storm's regular engine which performs model checking on a sparse-matrix representation of the MC generated from the BN. We now consider symbolic MTBDD-based model checking and compare this to state-of-the-art symbolic inference techniques on BNs.

10. One could alternatively add the hypothesis to the CNF formula as new clauses.

	Construction time (in s)	Inference time (in s)
`andes` - Storm, bisimulation	154.66	0.583
`andes` - Storm, MTBDD	303.15	avg: 3.298
`andes` - PSDD, minfill vtree	4.724	3.423
`win95pts` - Storm, bisimulation	0.149	0.002
`win95pts` - Storm, MTBDD	15.740	avg: 0.077
`win95pts` - PSDD, minfill vtree	0.047	0.017

Table 3: Symbolic MTBDD-based model checking vs. symbolic PSDD-based inference.

Baseline. As baseline, we use BN tools that use *probabilistic sentential decision diagrams (PSDD)* (Kisa et al., 2014), as described in Section 2.3. PSDDs are supported by two software packages; `psdd_nips` takes binary-valued BNs in `uai` format and compiles them into vtree and PSDD representations[11]. The inference is done on the psdd and vtree files using the `psdd` package[12]. Both packages were developed in Darwiche's group at UCLA.

RQ3: *How do bisimulation minimization techniques and symbolic model checking approach compare to PSDD-based inference?*

We performed a series of experiments to compute BN marginals using Storm's symbolic and bisimulation minimization techniques and using the PSDD packages. Since the PSDD packages are restricted to binary random variables, we considered the binary BNs `win95pts` and `andes`. Table 3 indicates the compilation and inference time (in seconds) for these benchmarks. We compiled the PSDD for all available vtree methods; i.e., *fixed branching factor*, *random branching factor*, and *minfill*. The table reports the timing for the *minfill* vtree that had the best performance. The inference time for bisimulation and for PSDD refers to the time for computing the marginal probabilities for all the variable valuations of the BNs, while for MTBDD the timing refers to the average time for computing all the probabilities. This is because computing all the MC state reachability probabilities in a single model-checking run is supported by Storm for the regular engine, but not (yet) for the MTBDD-based engine. Compiling the reduced MTBDD and computing the bisimulation quotients are performed by the built-in generic algorithms in Storm for MCs, this is not a BN-specific algorithm.

PSDDs are in general more succinct than MTBDDs (Bova, 2016) as they do not contain the multiplications of variable valuations, see Fig. 5 and 6. However, inference using Storm can be about one order of magnitude faster than PSDD using bisimulation. We refer to (Salmani & Katoen, 2020) for our more detailed experimental results. It is worthwhile to mention that Storm supports hypotheses involving multiple variables and arbitrary first-order logic formulas also for the symbolic and bisimulation engines. PSDD is—like Ace—restricted to hypotheses of single variables.

11. `https://github.com/hahaXD/psdd_nips`
12. `https://github.com/hahaXD/psdd`

6.3 Evidence-tailored Translation

All our experiments so far used the evidence-agnostic generation of MCs from BNs. We now consider MC generation that is tailored to the evidence, cf. Section 4.

RQ4: *What is the efficiency gain for evidence-tailored analysis?*

The implementation of the evidence-tailored translation prunes the BN with respect to the evidence and compiles the obtained sub-BN into a MC. This MC is analysed using symbolic model checking with MTBDDs. We conducted a series of experiments to investigate how the performance of inference is affected compared to the evidence-agnostic MC generation. We took the same topological order for both the evidence-tailored and the evidence-agnostic translation. The evidence nodes were randomly picked. To understand the influence of the evidence size (in terms of the number of BN nodes), we varied the fraction of BN nodes in the evidence.

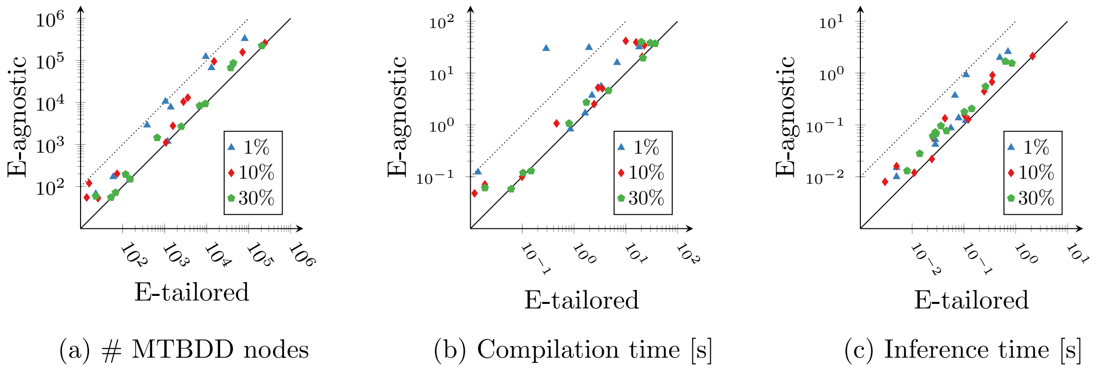


Figure 10: Evidence-tailored vs. evidence-agnostic BN2MTBDD.

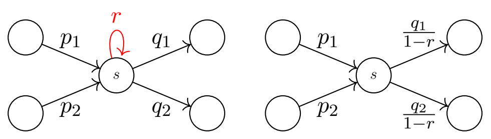
Figure 10(a) indicates the number of MTBDD nodes for the evidence-tailored versus the evidence-agnostic translation, (b) the MTBDD compilation time for both translations, and (c) the model-checking time for different number of evidence nodes (as fraction of the total number of BN nodes). All plots are log-log scale with the benchmarks from Table 1. The legend (e.g., 10%) indicates the ratio of the evidence size to the total number of BN nodes. The dots above the solid line indicate the evidence-tailored method outperforming the evidence-agnostic. The dashed-line indicates 1 magnitude difference. We observe that tailoring to the evidence gains up to about one order of magnitude if the fraction of BN nodes in the evidence is relatively low. This effect diminishes on increasing the size of the evidence. We expect that tailoring the topological order with respect to the evidence could yield a further improvement.

7. Parameter Synthesis Algorithms

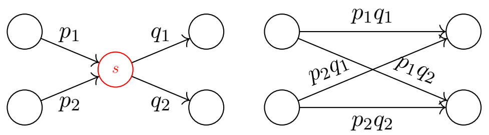
We now consider algorithmic techniques to determine the objectives on *parametric* MCs as described in Section 3.2: computing solution functions, feasibility checking, region verification, and parameter space partitioning. pMC parameter synthesis algorithms often exploit probabilistic model-checking techniques, e.g., verifying candidate instantiations in feasibility checking amounts to verify whether an MC satisfies a threshold on a reachability probability—a typical instance of probabilistic model checking. Another example is verifying the Markov decision process that is obtained from the relaxation of pMCs in parameter lifting (see below), an abstraction technique towards parameter partitioning.

7.1 Computing Solution Functions

One of the first problems considered for pMCs is to compute closed-form solutions, i.e., rational functions in the parameters. These functions map parameter values onto the reachability probabilities of interest. Let the state variable p_s denote the probability of reaching the target state t from the state s. The solution functions are obtained by solving an equation system in terms of the state variables and the parameters. Due to the occurrence of their products in the equations, the equation system is non-linear. Mathematically, the rational functions can be obtained by Gaussian elimination over the field of rational functions. Practically, an iterative *state elimination* procedure (Daws, 2004) is used that simplifies the pMC into a form with a single source state and the various target states of the reachability objective. This procedure is similar to computing regular expressions from non-deterministic finite automata (NFA) (Hopcroft, Motwani, & Ullman, 2003) and the performance similarly depends on the order in which the states are eliminated (Han, 2013). The key operations in pMC state elimination are *eliminating self-loops* and *adding shortcuts*, see Fig. 11(a) and (b), respectively. The former eliminates a self-loop at state s by re-scaling all its outgoing transitions. The latter replaces the transitions from s to t and from t to s' by a direct shortcut from s to s'.



(a) Elimination of self-loop.



(b) Elimination of states without self-loops.

Figure 11: State elimination in Markov chains.

7.2 Feasibility

For pMC $\mathcal{M}$ with parameters X and the constraint φ, the feasibility problem aims at finding an instantiation u for the parameters—if any—such that $\mathcal{M}[u] \models \varphi$[13]. Checking whether $\mathcal{M}[u] \models \varphi$ is possible by performing model checking on the obtained Markov chain, i.e., $\mathcal{M}[u]$ against the constraint φ or by instantiating the solution function $f_{\mathcal{M},\varphi}$ with u. What remains is how to pick the candidate instantiations. There are different techniques for that.

Particle Swarm Optimization (PSO). PSO is a sampling-based technique that attempts to solve a mathematical problem by iteratively selecting a candidate solution and moving in the search space towards the final answer. Applying PSO to feasibility analysis of pMCs includes

13. In this work we are more concretely interested in feasibility analysis for pMC reachability probabilities. Let G be the set of goal states in $\mathcal{M}$. Let $\lambda > 0$ be a probability threshold, and $\sim \in \{\leq, \geq\}$. The feasibility probability for reachability on pMCs is defined as follows: finding the instantiation u for the parameters, such that $Pr_{\mathcal{M}[u]}(\Diamond G) \sim \lambda$.

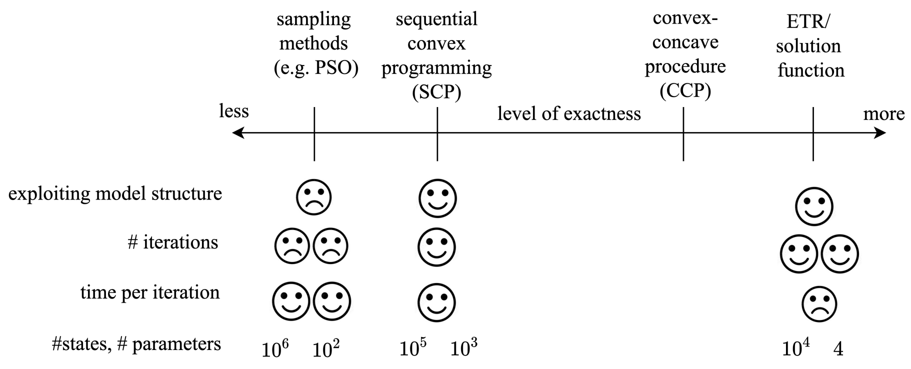


Figure 12: Overview on the practical approaches for the feasibility problem.

guessing an instantiation u (i.e., sampling), verifying whether $\mathcal{M}[u] \models \varphi$, and continuing the search until an accepting instantiation is found. The drawback is that the number of required iterations can easily get out of hand, and that the model structure of $\mathcal{M}$ is not exploited. Instead of taking the sampling-based approach to the feasibility problem, one can exploit SMT (Satisfiability Modulo Theory, where the theory is non-linear arithmetic over the reals) solvers to check the satisfiability of the corresponding ETR-formula, but such an approach can be very costly as solving ETR-formulas is in PSPACE.

Quadratically-Constrained Quadratic Program (QCQP). QCQP is a mathematical optimization problem that is exploited in pMC feasibility analysis (Cubuktepe et al., 2018) as a compromise between the pure sampling-based methods (PSO) and the exact, but computationally intensive—satisfiability methods. The idea is to initially guess a parameter instantiation, and then try to optimize around this guess in the parameter space, such that the objective function of interest—a reachability probability in our setting—is improved at each iteration. Such improvement aims at reducing the total number of iterations needed to find the accepting instantiation. However, it remains to ensure that each optimization step is computationally tractable and yields a correct solution. The key to this is to relax the ETR-formula and solve this, rather than solving the original formula. When all the transition probabilities are linear in the parameters—which is the case in the standard parameterization schema of BNs (Kjærulff & Van der Gaag, 2000)—the ETR-formula reduces to a QCQP (Boyd & Vandenberghe, 2004), an optimization problem with a quadratic objective and quadratic constraints. The software tool Prophesy (Dehnert et al., 2015) solves these problems by adopting a sequential-convex-optimisation (SCP) approach: it iteratively solves LPs (linear programs that relax the QCQP) to either find a solution of the QCQP (Chen et al., 2013; Puterman, 2014; Yuan, 2015) or find out that such solution does not exist. For further details, see Cubuktepe et al. (2018, 2022).

Figure 12 compares the sampling-based methods such as PSO, the SCP-based methods such as QCQP, and the exact SMT-based techniques. While the latter provides exact solutions, it is not salable to more than a handful of parameters. PSO and QCQP, on the other hand, may take several iterations to find the solution, but each of the iterations is computationally inexpensive. As QCQP directly exploits the model structure, it allows tackling pMC models with up to several hundreds of parameters.

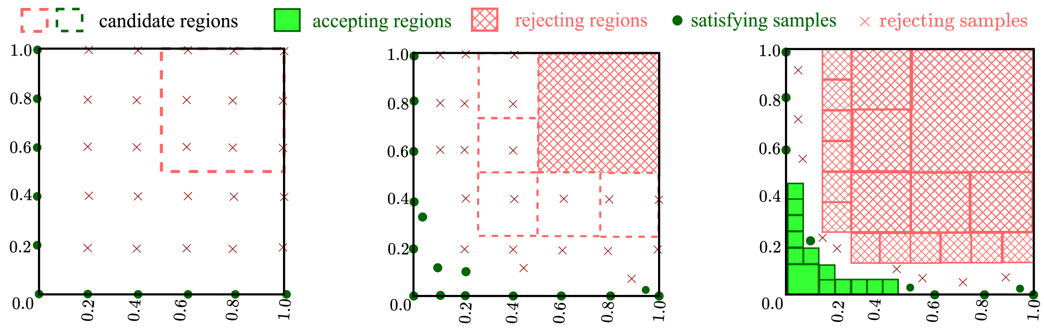
Stochastic Gradient-Descent (GD). GD is a first-order optimization method that maximizes an objective function by updating the parameters in the direction of its gradient. The term "first order" refers to the fact that the method works based on the first derivative of a function, and does not take the higher derivatives into account. GD has been recently successfully adopted for the feasibility problem (Heck et al., 2022). The main idea is to use the gradient of the reachability objective with respect to a parameter p, say, at some (initially guessed) instantiation u, and then apply the hill-climbing property of stochastic GD techniques to find a feasible instantiation. The key step to enable GD methods is to efficiently compute the gradient. Various GD update methods—including Plain GD, Momentum GD (McClelland et al., 1986), Nesterov accelerated GD (Sutskever et al., 2013), RMSProp (Sutskever et al., 2013), Adam (Kingma & Ba, 2015), and RAdam (Liu et al., 2020)—are taken to tackle the feasibility problem (Heck et al., 2022). Since Adam has been experimentally shown to be one of the most efficient methods in pMC feasibility analysis (Heck et al., 2022), we took that as the basis.

7.3 Parameter Space Partitioning

Let $\mathcal{M}$ be a pMC over the set X of parameters, φ a constraint, $R \subseteq \mathbb{R}^n$ a region, and u a parameter instantiation of X. Recall that for a coverage factor $0 \leq \eta \leq 1$, the parameter space partitioning problem aims at partitioning the region R into an accepting fragment R_+, a rejecting fragment R_-, such that the remaining inconclusive fragment $R_?$ covers at most $1-\eta$ of the size of R. The partitioning is *exact* if η=1, i.e., $R_? = \emptyset$.

The approach to tackle the parameter space partitioning problem is inspired by the principle of *counter-example guided abstraction refinement* (CEGAR) (Clarke, Grumberg, Jha, Lu, & Veith, 2000), a successful technique in model checking. In our setting, the region R is successively divided into accepting and rejecting (rectangular or quads) regions. The idea is to compute a sequence of rectangular accepting regions that successively extend each other. Similarly, at each iteration, the sequence of rejecting regions is extended. Let $\left(R^i_+\right)$ denote the sequence of accepting regions and $\left(R^i_-\right)$ denote the sequence of rejecting regions. At the i-th iteration, $R^i = R^i_- \cup R^i_+$. The iterative procedure halts when R^i covers at least $100{\cdot}\eta\%$ of R. A solution to the exact synthesis problem is obtained in the limit as $\lim_{i\to\infty} R^i_+ = R_+$ and $\lim_{i\to\infty} R^i_- = R_-$.

Picking region candidates. An essential issue for parameter space partitioning is how to pick good region candidates. One strategy is to initially start with the entire region R, but that is typically large and inconclusive. An alternative is to first do some *sampling* and split the region accordingly. Sampling refers to picking an instantiation u and verifying using probabilistic model checking whether $\mathcal{M}[u] \models \varphi$. Splitting the regions at each iteration can be done according to different strategies. Examples are splitting the regions uniformly into equally-sized regions or starting with smaller regions and gradually moving towards larger candidates. In practice, the regions are initially preferred when the verification seems less costly: candidate regions that are seemingly accepting (according to the sampling results)

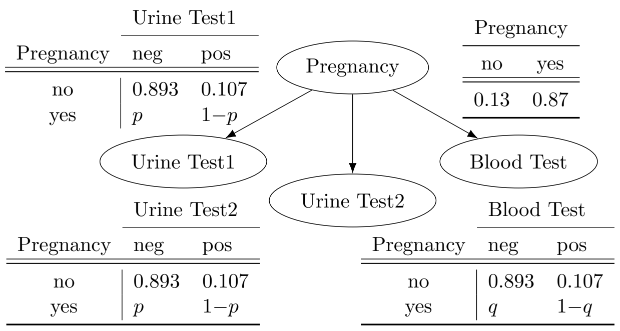


(a) Uniform Sampling. (b) Verifying subregions. (c) Preliminary result.

Figure 13: Parameter space partitioning in progress.

and are far from rejecting samples or regions that are preferred over those regions which have rejecting samples or are close to rejecting regions.

The process is illustrated in Fig. 13. Partitioning starts with (a) uniformly sampling the entire parameter space, e.g., at grid points. Guided by the sampling results, (b) the region candidates are picked and verified. The procedure iteratively continues: the inconclusive regions are refined and gradually become smaller. Figure 13(c) depicts an intermediate result of the iterative partitioning. The white areas in the middle are still inconclusive: they contain both accepting instantiations (green) and rejecting ones (red).

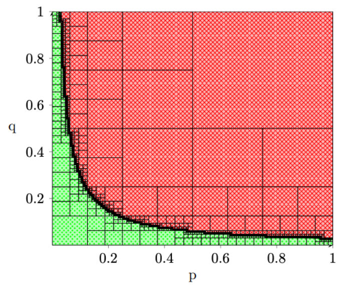


	Urine Test1	
Pregnancy	neg	pos
no	0.893	0.107
yes	p	$1-p$

Pregnancy	
no	yes
0.13	0.87

	Urine Test2	
Pregnancy	neg	pos
no	0.893	0.107
yes	p	$1-p$

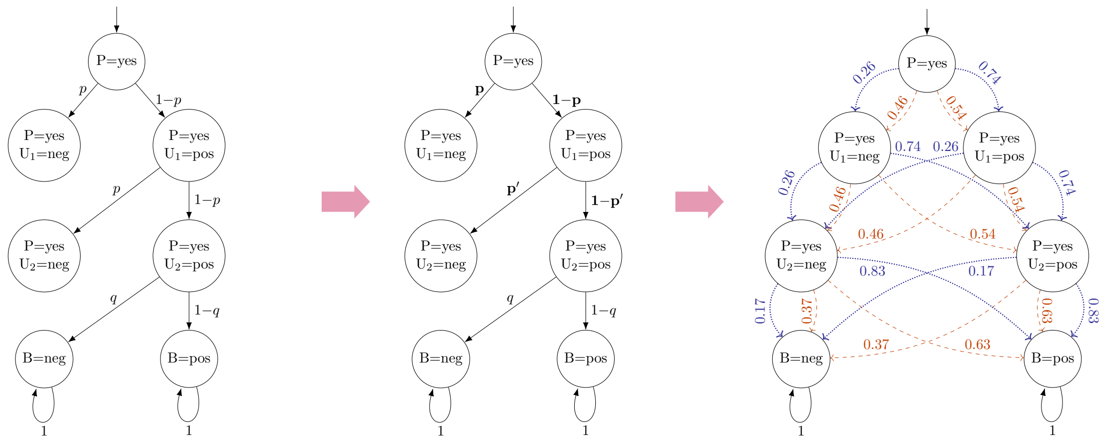
	Blood Test	
Pregnancy	neg	pos
no	0.893	0.107
yes	q	$1-q$



(a) The pBN with unknown probabilities. (b) Parameter space partitioning.

Figure 14: An example pBN with parameter dependencies - Pregnancy tests.

Example 17. Figure 14(a) depicts a variant of our running example with parameter dependencies: the farmer uses the urine test twice for each cow to detect pregnancy; parameter p occurs in two different rows of two different CPTs. Figure 14(b) indicates the result of approximate parameter partitioning for η=0.99 and the specification $\Pr(\mathit{Pregnancy} = \text{yes} \mid \mathit{Urine\ Test1} = \text{neg} \wedge \mathit{Urine\ Test2} = \text{neg} \wedge \mathit{Blood\ Test} = \text{neg}) \leq 1/5$. The green areas denote the satisfying regions and the red areas depict the rejecting regions. The black areas in the middle represent the inconclusive regions and occupy at most 1% of the entire parameter space.



(a) The original (sub-)pMC. $0.26 \leq p \leq 0.46, 0.17 \leq q \leq 0.37$.

(b) The pMC with no parameter dependencies.

(c) Non-parametric MDP.

Figure 15: Parameter lifting algorithm (PLA) steps: (a) the parameteric MC $\xrightarrow{\text{relaxation}}$ (b) the pMC with no parameter dependencies $\xrightarrow{\text{substitution}}$ (c) non-paramertric MDP.

7.4 Region Verification

A key step to (approximately) partition the parameter space R is to verify the sub-regions of R against the constraint φ. An exact approach is to compute the ETR-formula and exploit an SMT checker (for non-linear real arithmetic) to check whether the ETR-formula conjuncted with the region formula (e.g., $3/10 < p < 6/10$) is satisfiable. Such an approach is exact but costly. Parameter lifting (Quatmann et al., 2016) is an over-approximation method to region verification. It reduces the region verification problem on a pMC to model checking queries on a relaxed *non-parametric* Markov Decision Process.

Definition 3. A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, s_I, Act, \mathcal{P})$ with a finite set S of states, an initial state $s_I \in S$, a finite set Act of actions, and a (partial) transition probability function $\mathcal{P} : S \times Act \to Distr(S)$.

The relaxation is done by first eliminating the parameter dependencies in the pMC and then replacing the transition probabilities with the extremal (maximal and minimal) values of the parameter interval. The obtained MDP over(under)-approximates the reachability probabilities of the original pMC: if the MDP satisfies the constraint for all the schedulers, the region is accepting for the original pMC. If the MDP satisfies the negation of the constraint, the region is rejecting. This enables solving the region verification as a probabilistic model-checking problem.

Example 18. Figure 15 demonstrates the parameter lifting steps. The (a) pMC is from the evidence-agnostic pMC of our pBN example with two urine tests; see Example 17, Fig. 14 (a). We show only a subpart of the pMC that is relevant to demonstrate parameter lifting algorithm. Parameter p is repeated in the outgoing transitions of two distinct states. The *relaxation* step equips the second state (P=yes, U_1 = pos) with a fresh parameter, yielding a pMC free of parameter dependencies; see Fig. 15(b). What remains is to compute the extremal values for each parametric function. Consider the region R_{sub} with $p \in [0.26, 46], q \in [0.17, 0.37]$. The extremal (minimal and maximal) values for e.g., parameter p are 0.26 and 0.46 for the function p (and p') and 0.54 and 0.74 for the function $1-p$ (and $1-p'$). In the *substitution* phase, the local parametric choices at each state are replaced by a *non-deterministic* choice between these extremal values. This yields a parameter-free MDP, see Fig. 15(c).

Ratio and difference parameter tuning. Let us describe how parameter lifting can be exploited to the ratio and difference parameter tuning problems on parametric BNs. We explain this for $\sim = \geq$; the approach can easily be adapted for $\leq$. The *ratio* problem on pBN $\mathcal{B}$ reduces to finding an instantiation u in the pMC $\mathcal{M}^{\varrho}_{\mathcal{B},E}$ such that:

$$\varphi_{\text{ratio}}\colon \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}[u](\Diamond\text{num}) \;\geq\; \lambda \cdot \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}[u](\Diamond\text{denom}), \tag{6}$$

where $\Pr(\Diamond\text{num})$ stands for $1 - \Pr[u](\Diamond(H = \neg h' \vee E = \neg e))$ and $\Pr(\Diamond\text{denom})$ abbreviates $1 - \Pr[u](\Diamond(H = \neg h \vee E = \neg e))$. Note that φ_{ratio} is obtained from equation (2) and Proposition 1. The ratio problem can then be solved using parameter lifting in the following way: let region $R \subseteq \mathbb{R}^n_{\geq 0}$ where n is the number of parameters in the pBN. We perform parameter lifting for reaching *num* and reaching *denom* on $\mathcal{M}_{\mathcal{B},E}$, respectively. This gives upper (UB_{num}, UB_{denom}) and lower bounds (LB_{num}, LB_{denom}) for the probabilities on the left- and the right-hand side of equation (6). Verifying region R against constraint φ_{ratio} reduces to checking whether $[LB_{\text{num}}, UB_{\text{num}}] \geq \lambda \cdot [LB_{\text{denom}}, UB_{\text{denom}}]$:

- If $LB_{\text{num}} \geq \lambda \cdot UB_{\text{denom}}$, the region R is accepting for the ratio property.
- If $UB_{\text{num}} \leq \lambda \cdot LB_{\text{denom}}$, the region R is rejecting for the ratio property.
- Otherwise, refine the region R.

For *difference* parameter tuning, we adopt the above recipe by replacing equation (6) by:

$$\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}[u](\Diamond\text{num}) \;\geq\; \lambda + \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}[u](\Diamond\text{denom}).$$

8. Experimental Results: The Parametric Setting

Our pBN tool-chain (see Fig. 1 on page 1639) implements the pBNs to pMC transformation and imposes pBN objectives as PCTL formulas on pMCs. We use both the tools Storm and Prophesy (Dehnert et al., 2015). The pBNs are expressed in an extended version of the `bif` format and the pMCs are either encoded in Jani or `drn`. Our tool also supports transforming non-parametric BNs into MCs and parametrizes the MC with respect to the BN CPT entries. This enables performing sensitivity analysis and parameter synthesis on pBNs using pMC techniques. Our tool uses Storm to compute the pBN sensitivity functions. For parameter tuning of pBNs, it exploits parameter lifting to partition the parameter space into satisfying, refuting, and inconclusive areas. The size of the unknown area depends on the approximation factor η (see Section 5). Storm and Prophesy are

	SamIam	Bayesserver	Storm-Prophesy
computing sensitivity function	✗	$p_{\leq 2}c_{\leq 2}r_1$	$p_*c_*r_*$
computing sensitivity value	✗	$p_{\leq 2}c_{\leq 2}r_1$	$p_1c_*r_*$
simple parameter tuning	$p_*c_1r_1$	$p_1c_1r_1$	$p_*c_*r_*$
difference parameter tuning	$p_*c_1r_1$	$p_1c_1r_1$	$p_*c_*r_*$
ratio parameter tuning	$p_*c_1r_1$	$p_1c_1r_1$	$p_*c_*r_*$

Table 4: Overview of the capabilities of the pBN synthesis tools.

exploited for feasibility checking: finding a parameter instance that satisfies an inference query. We have pursued the following different techniques for feasibility analysis: (1) particle swarm optimization (PSO), (2) mathematical optimisation using quadratically-constrained quadratic programming (QCQP), and (3) stochastic gradient-descent (GD). The first two techniques are supported by Prophesy, the last one by Storm. For QCQP experiments, we used the latest package on top of Prophesy that offers recent techniques to solve QCQP for parametric Markov chains (Cubuktepe et al., 2022). Our tool-chain supports pBNs of the $p_*c_*r_*$ class, that is no restrictions are imposed on the number of parameters nor on the number of CPTs in which they occur.

Baseline. For the evaluation of our tool to analyze pBNs, we used as baseline two synthesis tools for parametric BNs: *Bayesserver*[14] and *SamIam*[15]. *Bayesserver* is a commercial tool that offers sensitivity analysis and parameter tuning of pBNs. For sensitivity analysis, it computes the sensitivity function and sensitivity value. It also performs minimal-change parameter tuning for conditional, hypothesis ratio, and hypothesis difference constraints. Bayesserver is restricted to the pBN subclasses $p_1c_1r_1$ and $p_2c_{\leq 2}r_1$ for sensitivity analysis and to the subclass $p_1c_1r_1$ for parameter tuning. *SamIam* is a tool for the sensitivity analysis on pBNs, developed at Darwiche's group. It allows for the specification of conditional, hypothesis ratio, and hypothesis difference constraints as described in Section 3.1. SamIam aims to identify minimal parameter changes that are necessary to satisfy the specified constraints. It supports the pBN classes $p_1c_1r_1$ and $p_*c_1r_1$. Table 4 summarises the features of all tools used for pBNs in our experiments and indicates for which classes of pBNs their features are supported.

8.1 Computing Sensitivity Functions

RQ5: *Can we perform sensitivity analysis on pBNs faster than Bayesserver and scale to pBNs with more parameters and arbitrary parameter dependencies?*

One of the pivotal tasks for sensitivity analysis on pBNs is to compute the sensitivity functions. For this task, we performed a series of experiments on various benchmarks. The experiments are categorized for the pBN subclasses p_1, p_2, and p_n for $n > 2$, where the subscript denotes the number of parameters in the pBN.

14. `https://www.bayesserver.com`
15. `http://reasoning.cs.ucla.edu/samiam`

Figure 16 presents the results in a log-log scale plot with timings in seconds. The blue and the red (triangle and diamond) points correspond to the experiments with one and two parameters: the same pBN and the same query were taken and the corresponding sensitivity function was computed by Storm—using the bisimulation minimization technique applied to pMCs—and by Bayesserver. The results indicate that for most of the cases, Storm computes the solution functions faster (up to one order of magnitude) than Bayesserver. The pentagon points denote the experiments for the p_* subclass; i.e.; the pBNs with multiple parameters and the parameters occurring in multiple CPTs. Bayesserver does not support p_* instances; see the top-most horizontal NS (not supported) line in the plot. Storm computed the sensitivity functions for the p_* instances up to 200 and 380 parameters respectively for large pBNs such as `win95pts` and `hailfinder` in about 400 and 30 minutes, respectively.

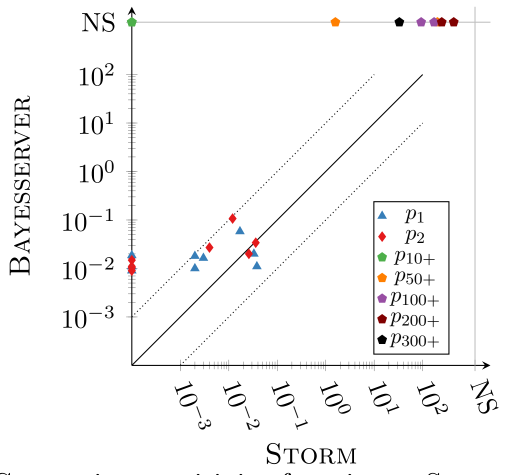


Figure 16: Computing sensitivity functions - Storm vs. Bayesserver.

Computing solution functions is a computationally hard problem. It is exponential in the number of parameters, and polynomial in the number of pMC states and the maximal degree of the polynomial transition probability functions (Hutschenreiter, Baier, & Klein, 2017; Baier, Hensel, Hutschenreiter, Junges, Katoen, & Klein, 2020)[16]. Our next research question is aimed to investigate the practical complexity of pBN sensitivity analysis using pMC techniques.

RQ6: *What are the decisive factors in the computation time of pBN sensitivity functions using PMC techniques?*

To address this question, we computed sensitivity functions for a set of `bnlearn` benchmarks and analyzed the obtained functions. For a given BN and a given query, we incrementally increased (in steps of five) the number of parameters in the ancestors of query nodes, and calculated the sensitivity function for the resulting pBN.

Table 5 lists our experimental results with the maximal number of handled parameters for each pBN. The columns refer to the name of the pBN, its number of parameters, the time

16. Our pBNs are parameterized according to the standard schema for BN parameterization that keeps the linear ratio of the original probability entries in each CPT row. Hence, the degree of the transition probability functions in the pMCs is one.

network	#params	time [s]	#states	#transitions	degree	#summands
`child`	**60**	269	**587**	**1,909**	13	6,529,728
`alarm`	85	201	1,118	3,174	10	651,937
`hepar2`	135	173	11,698	32,289	12	3,855,904
`insurance`	140	83	22,740	71,074	7	2,780,076
`win95pts`	200	**437**	9,947	17,948	**16**	**7,529,772**
`water`	255	247	**37,569**	**72,778**	9	273,011
`hailfinder`	**380**	**33**	20,304	55,976	**3**	**9,943**

Table 5: The statistics on computing pBN sensitivity functions.

for computing the sensitivity function (in seconds), and the number of states and transitions of its pMC. The last two columns refer to the degree of the obtained sensitivity function and the total number of summands in its numerator and denominator. The rows are sorted by the number of parameters and the minimum and maximum numbers in each column are highlighted in boldface. The results suggest a strong correlation between the computation time and the characteristics of the sensitivity function; e.g., `win95pts` and `hailfinder` are the networks with the highest and lowest computation times and this directly correlates to their degree and number of summands.

This effect is also visualized in Figure 17. The x-axis lists the pBN benchmarks, the left y-axis indicates the computation time in seconds, and the right y-axis indicates sensitivity function characteristics such as the degree of the sensitivity function and the (logarithm of the) number of summands. The benchmarks along the x-axis are sorted by their number of parameters; nevertheless, the computation time fluctuates. The most strongly correlated factor to the computation time seems to be the degree of the obtained sensitivity function.

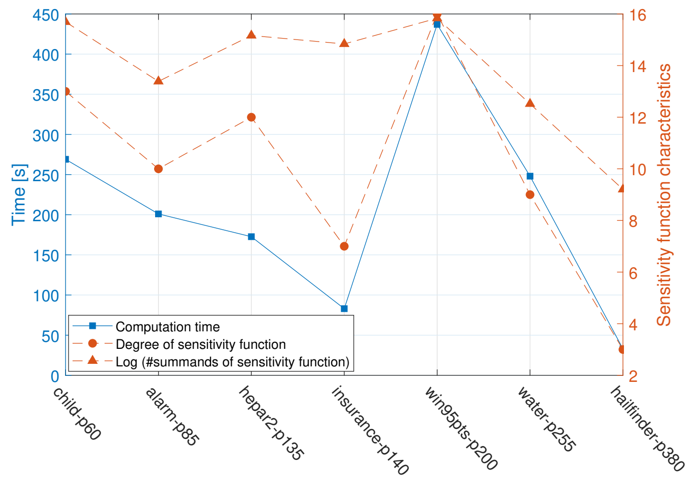


Figure 17: Characteristics of the pBN sensitivity functions and their computation time.

From the pBN perspective, this degree is related to the number of CPTs that are parameterized; in the `hailfinder` benchmarks the parameters occur in only three different CPTs with many entries; in `win95pts`, 22 CPTs contain parameters. Parameters occurring in many dependent CPTs yield a large number of monomials. This leads to the worst-case complexity of the problem that is exponential in the number of parameters (Hutschenreiter et al., 2017; Baier et al., 2020).

Our experiments indicate that Storm computes pBN sensitivity functions mostly faster than the baseline tool Bayesserver and for a substantially larger class of pBNs: there is no limit on the number of parameters, how often a parameter occurs in a CPT, nor on the number of parameterized CPTs. We extended our experiments to measure the characteristics of the obtained sensitivity function. While the number of pMC states has a linear influence on the computation time, the most correlated factor is the degree of the obtained sensitivity function that is related to the number of parameterized CPTs.

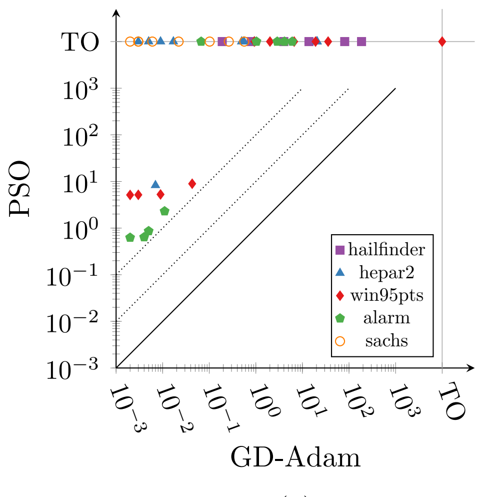
8.2 Feasibility Analysis

Parametrization and setup. We took various BN benchmarks from the `bnlearn` repository and considered for each benchmark various parametric instances. We gradually increased the number of parameters in each BN, starting from the hypothesis nodes and continuing to their parents and ancestors. The parameters in the $(i{+}1)$-st instance extend those in the i-the instance. The hypothesis in the objective is the disjunction of the variable valuations of all BN leave nodes. This is aimed at queries that are sensitive to many CPT entries. Since the probability of such disjunctions of multiple variable valuations tends to be large, we used $\leq$ as the comparison operator in the query. Note that the satisfying instantiation u found for the threshold λ_1 also respects any threshold $\lambda_2 \geq \lambda_1$. We therefore aimed at finding restrictive constraints and did so by computing the minimal values for each pBN instance using GD while using a 10% relaxation of the minimal probability as the threshold.

GD Setup. Various GD update methods—including Plain GD, Momentum GD (McClelland et al., 1986), Nesterov accelerated GD (Sutskever et al., 2013), RMSProp (Sutskever et al., 2013), Adam (Kingma & Ba, 2015), and RAdam (Liu et al., 2020)—are implemented in Storm (Heck et al., 2022). Since Adam has been experimentally shown to be one of the most efficient methods in pMC feasibility analysis (Heck et al., 2022), we took that as the update method for our feasibility experiments. Adam is an adaptive algorithm in which the learning rate changes over time and each parameter has its own learning rate; parameters with larger gradients have smaller learning rates than the ones with smaller gradients. The following GD constants in Storm are set according to the standard settings in the literature (Kingma & Ba, 2015; Ruder, 2016; Liu et al., 2020): the batch size is 32, the average decay is 0.9, and the squared average decay is 0.999.

QCQP Setup. For QCQP experiments, we used the latest package [17] on top of Prophesy that supports both convexification methods to solve QCQP: *Sequential Convex Programming* (SCP) (Cubuktepe et al., 2022) and *Convex Concave Procedure* (CCP) (Cubuktepe et al., 2018).

17. `https://zenodo.org/record/5745851#.Y_9XKuzMK3I`



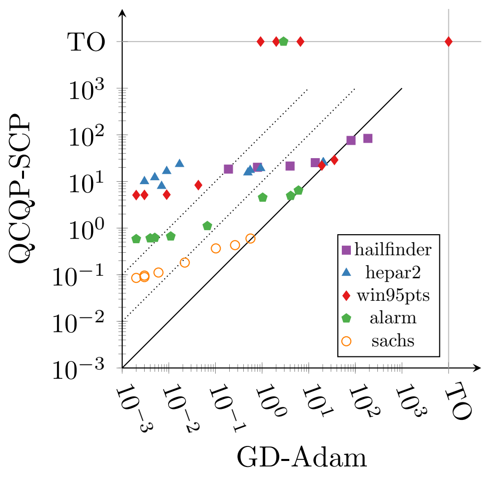


Figure 18: Feasibility analysis on pBNs: GD vs. PSO (a) and QCQP (b).

RQ8: *How effective are the pMC feasibility checking techniques for pBN analysis?*

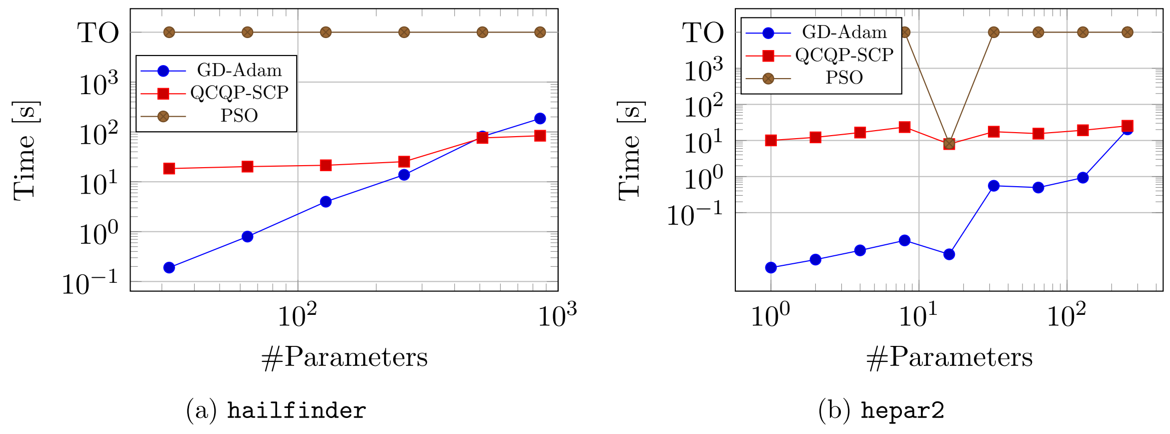
We exploited three pMC feasibility checking methods on the pBN benchmarks: GD-Adam, QCQP, and PSO. The aim is to find a parameter instantiation u for pBN $\mathcal{B}$ such that the BN $\mathcal{B}[u]$ satisfies the given constraint, i.e., the inference query φ, the comparison operator $\sim$, and the threshold λ. Figure 18(a) displays the results of PSO vs. GD-Adam, while Fig. 18(b) displays QCQP vs. GD-Adam for various instances of each network. We did not separately plot the results of QCQP-SCP and QCQP-CCP as the they were mostly overlapping [18]. The plots are in log-log scale and each axis indicates the feasibility analysis time (in seconds) taken by the corresponding method. The time-out is 15 minutes.

We observe that QCQP is more effective than PSO: QCQP encountered a time-out in five cases, while PSO encountered a time-out in 34 cases. All cases, except for one instance of `win95pts`, were handled by GD within 100-200 seconds. In general, the experimental results indicate that GD substantially outperforms both PSO and QCQP, often by one or two orders of magnitude. However, in several cases QCQP is faster than GD in finding a feasible instantiation. This occurs for pBN instances with many (more than 200) unknown parameters. See RQ9 and Tab. 6.

RQ9: *How do the number of parameters influence the feasibility analysis time?*

We investigated how the number of unknown parameters in the pBN instances affects the feasibility analysis time. Figure 19 indicates the results of the feasibility analysis experiments for four of our benchmarks in detail using a log-log plot. The x-axis indicates the number of parameters, whereas the y-axis indicates the time in seconds. The feasibility checking techniques scale up to 853 parameters for `hailfinder`, our pBN benchmark with the highest number of parameters. The analysis time in general grows on increasing the number of parameters. Gradient-descent is, however, the most sensitive to the number of

18. We distinguish QCQP-CCP and QCQP-SP in Table. 6



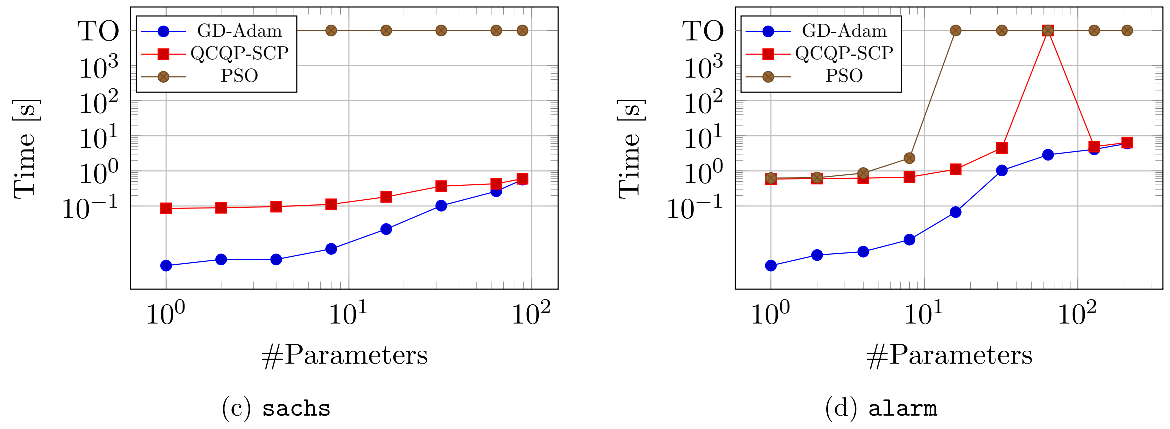


Figure 19: Feasibility analysis on pBNs using pMC methods PSO, QCQP, and GD-Adam: the analysis time as the number of unknown parameters in the network is increased.

parameters. This comes from the fact that GD has to compute the gradient for each parameter, while QCQP techniques benefit from the iterations of sampling, convexifying, and model checking, and are not highly affected by the number of parameters.

Table 6 gives a more detailed overview on the instances with more than 200 parameters. The first two columns show the information on parametric BN: the name of the BN and the number of unknown parameters. The two successive columns indicate the pMC info including the number of states and the number of transitions. The table reports the feasibility checking time (in seconds) for PSO, QCQP, and GD. For QCQP, we report the results from both CCP and SCP Convexification methods. For each method, the time (in seconds) and the number of iterations is indicated. The best result in each row is highlighted in boldface.

We observe for the pBN instances with a high number of parameters that QCQP beats GD: gradient-descent solves the problems within $\sim$ 200 seconds, while QCQP techniques handle them within $\sim$ 100 seconds.

pBN info		pMC info		PSO	QCQP				GD
					SCP		CCP		Adam
network	**# par.**	# stat.	# trans.	**t**[s]	**t**[s]	iter	**t**[s]	iter	**t**[s]
`alarm`	210	1118	3174	TO	6.42	3	**4.01**	0	6.03
`win95pts`	256	9947	17948	TO	21.9	2	27.3	4	**19.1**
`win95pts`	326	9947	17948	TO	**29.0**	3	46.4	7	35.6
`hepar2`	256	11698	32289	TO	25.4	2	**18.5**	0	20.8
`hailfinder`	256	20304	55976	TO	25.3	2	23.5	0	**13.8**
`hailfinder`	512	20304	55976	TO	**75.8**	0	89.7	3	80.9
`hailfinder`	853	20304	55976	TO	**83.8**	1	**83.8**	1	185.8

Table 6: Detailed feasibility results: $\#parameters > 200$.

8.3 Parameter Space Partitioning

Our tool-chain adopts Storm's approximate parameter space partitioning for the task of parameter tuning on pBNs. Figure 20 visualizes the results on the `alarm` network with two parameters repeatedly occurring in 26 rows of three distinct CPTs. The left figure indicates the parameter space partitioning using Storm using parameter lifting algorithm with coverage $\eta = 0.99$. The green areas correspond to the regions satisfying the constraint, the red areas are the regions violating the constraint, and the black areas in the middle are the inconclusive regions. The right figure indicates the visualization of the solution function, where the vertical axis denotes the value of the solution function. The figures indicate that the parameter tuning problem on pBNs does not necessarily yield rectangular solution spaces. In other words, the simple single instantiations or simple inequalities do not provide the complete solution for pBN tuning constraints in the general case.

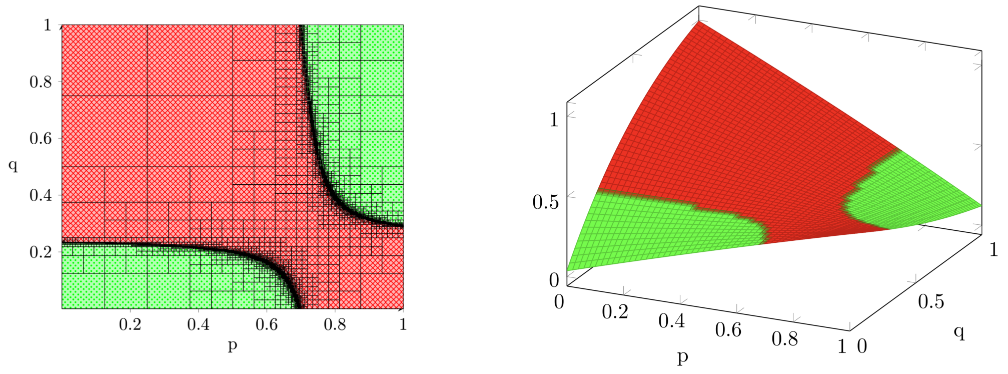


Figure 20: Parameter space partitioning for $\texttt{alarm-p}_2\texttt{c}_3\texttt{r}_{26}$.

RQ10: *How does the coverage affect the parameter space partitioning time for pBNs?* We performed a series of experiments on multiple pBNs including three pBN instances from the benchmarks `win95pts`, `hailfinder`, and `hepar2`, where each pBN included *eight* parameters. The parameters were in the CPT of the constraint hypothesis node. For our experiments, we incremented the coverage factor (η) in steps of 0.1 and measured the partitioning time as well as the fraction of the inconclusive area. Figure 21 (in log-log

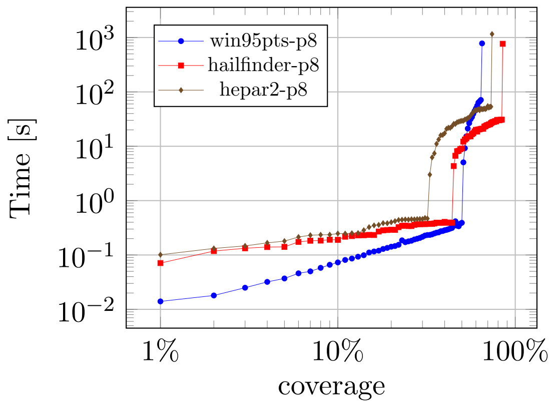


Figure 21: Parameter lifting on an 8-dimensional parameter space.

scale) indicates the results. The x-axis indicates the coverage percentage that is $100 \cdot \eta$, while the y-axis refers to total analysis time (in seconds). The time-out was 15 minutes.

Storm was able to cover 65% of the parameter space for `win95pts-p8` in 780.64 seconds, 74% of the parameter space for `hepar2-p8` in 1157.9 seconds, and 85% of the parameter space for `hailfinder-p8` in 762.26 seconds. The analysis time significantly drops when compromising on the coverage: 64% of the parameter space for `win95pts-p8` was covered in 71.07 seconds, 73% of the parameter space for `hepar2-p8` was covered in 53.45 seconds, and 84% of the parameter space for `hailfinder-p8` was covered in 33.66 seconds.

Table 7 gives the detailed results for `win95pts-p8`. The first column indicates the coverage factor; the next three columns list the percentage of accepting areas, rejecting areas, and the unknown areas respectively. The fifth column refers to the number of verified regions and the last two columns indicate the analysis time (in seconds) and the peak memory usage (in Mb). At η=1/2, half of the parameter space is covered and is rejecting. As the coverage factor increases, the analysis time increases, yielding a larger fragment of accepting and rejecting regions to be found. The parameter space partitioning time is tightly correlated with the number of regions to be verified. The Pearson correlation coefficient between the number of regions and time for `win95pts-p8` is 0.993.

Comparison to SamIam. Parameter space partitioning is a feature that seems not directly supported by existing tools for parametric BNs. SamIam can handle single parameters; it provides its solutions in form of single parameter suggestions (as simple inequalities, such as $p \geq 0.273$) and single-CPT parameter suggestions (as single instantiation points for the CPT entries). Parameter space partitioning, on the other hand, aims at partitioning the entire parameter space. It provides a set of accepting areas for a given constraint and provides more analysis information. A direct comparison between Storm and SamIam is thus not practicable. However, we ran SamIam on `win95pts` network to investigate where the reported CPT instantiation by SamIam lies within our partitioning results. The corresponding entry is highlighted in green in Tab. 20. Storm's accepting area (R_+) includes the instantiation suggested by SamIam for η=0.52, i.e., after covering 52% of the parameter space and only having determined 15% of the accepting area that a 65% coverage (last row) reveals.

coverage	accepting(%)	rejecting(%)	unknown(%)	**#regions**	**time**	mem
0.50	0	50	50	32896	0.4	95
0.51	0.92	50.08	49	432,991	5.1	641
0.52	1.47	50.53	48	808,606	9.2	1170
0.53	2.24	50.76	47	1,309,681	15.2	1908
0.54	2.85	51.15	46	1,705,186	21.2	2422
0.55	3.57	51.43	45	2,118,796	26.6	3133
0.56	4.29	51.71	44	2,560,456	31.5	3669
0.57	4.83	52.17	43	3,004,156	34.4	4202
0.58	5.67	52.33	42	3,397,366	39.4	4681
0.59	6.22	52.78	41	3,788,026	44.5	5158
0.60	7.02	52.98	40	4,264,621	49.5	6085
0.61	7.63	53.37	39	4,640,236	55.7	6516
0.62	8.32	53.68	38	5,083,936	63.7	6898
0.63	9.04	53.96	37	5,478,421	67.0	7387
0.64	9.64	54.36	36	5,968,531	71.1	8140
0.65	9.93	55.07	35	33,408,826	780.6	10991

Table 7: Some detailed results of parameter space partitioning for `win95pts-p8`.

9. Related Work

Bayesian network analysis using verification. As BNs are in some sense a simple form of (acyclic) probabilistic programs, in principle verification and other formal analysis techniques for probabilistic programs can be exploited to analyze BNs.

Claret et al. (2013), e.g., use data-flow analysis for exact inference of BNs and claim a higher precision than BN techniques.

Batz et al. (2018) map BNs with a given inference query onto probabilistic programs under a rejection sampling semantics. By exploiting specific properties of the resulting looping programs, they provide a closed-form solution for exact inference as well as for determining the expected run-time of the program until generating an i.i.d.-sample. This enables e.g., to automatically determine whether the expected sampling time is infinite due to conditioning on zero-probability evidences.

Bartocci et al. (2020, 2022) take a similar approach and encode their programs as prob-solvable loops. They obtain linear recurrences over statistical moments from the programs and use them to analyze BN objectives such as exact inference, sensitivity analysis, and filtering using algebraic reasoning. The sensitivity analysis boils down to computing symbolic functions similar to the solution functions in this paper. Parameter synthesis problems such as feasibility, tuning, and parameter space partitioning are not covered.

Deininger et al. (2016) apply MTBDD-based PMC to the analysis of factored probabilistic graphical models such as dynamic BNs. Their evaluation concludes that MTBDD-based model checking is in general not beneficial as the MTBDD size also depends on the number of different terminals they have (which influences sharing) and this is not reduced in factored representations. This is in line with our findings on comparing MTBDD versus PSDD.

Shih et al. (2018) take a different approach and directly use BDD-based techniques to formally verify BN classifiers. Holtzen et al. (2021) take a reverse approach: they investigate to what extent recent advances in probabilistic inference can be used to accelerate PMC of finite-horizon objectives (i.e., reachability within a given number of steps). They symbolically represent all horizon-length paths through the Markov chain and show better scalability compared to PMC tools on selected benchmarks.

We are unaware of any works on pursuing MC parameter synthesis algorithms for parametric BNs.

Bayesian inference using model counting. Weighted model counting (WMC) is an established method for probabilistic inference; see (Darwiche, 2002; Sang et al., 2005; Chavira & Darwiche, 2005, 2006, 2008; Bart et al., 2016; Sharma et al., 2019). More recently, Dilkas and Belle (2021) reflected on the existing encoding techniques for WMC and proposed a new method to encode conditional weights on literals corresponding to conditional probabilities. While this improves the performance is several instances compared to the previous encoding techniques, they conclude that the efficiency of the encoding mainly depends on the properties of the Bayesian networks at hand. Jerrum et al. (1986) originally addressed the theoretical foundations for approximate model counting for approximate inference. This direction was later followed by e.g., Emron et al. (2013, 2014), Achim et al. (2016), Chakraborty et al. (2013, 2016) and others. The techniques rely on (a) projecting high-dimensional probability distributions into lower dimensional spaces and (b) using hashing functions. Chakraborty et al. (2016) et al. e.g., exploit word-level hashing functions that enable using sophisticated SMT solvers.

Parametric Bayesian networks. Castillo et al. (1995) have originally investigated parametric BNs. They adapted numerical propagation algorithms for symbolic computations and noted that the symbolic part of the computations exponentially grows with the number of parameters. This work was followed by (Castillo et al., 1996, 1997a), where the authors focused on computing the query-relevant parameters and used the sensitivity functions to get upper and lower bounds for the marginals. (Castillo, Gutiérrez, Hadi, & Solares, 1997b) extended those methods to Gaussian BNs.

Chan & Darwiche (2000) proposed a counterpart method for BN sensitivity analysis. The method is inherently based on the joint-tree algorithm and was driven in two directions: one-way analysis, addressing the effect of single parameters, and n-way analysis, addressing the joint effect of n parameters. They focused n-way analysis on the case where all the parameters happen in the same clique of the junction tree.

Later in (2002), the same authors considered the task of one-way sensitivity analysis and parameter tuning in BNs. They extended the technique to multiple parameters in (2004), where the main focus was on the parameters all occurring in the same CPT. Such restriction was shown to make the problem computationally easier than the general n-way sensitivity analysis, keeping it equally hard as one-way for their proposed method. They proposed a distance measure in (2005) with the aim for bounding the deviation from the original values of the parameters in the network when tuning the parameters.

Laskey (1995) introduced a measurement to quantify the effect of changing parameters on the probability of interest. Van der Gaag & Renooij (2001) proposed another measurement to preserve the most likely outcome of the network. They indicate the effectiveness of their analysis method on a moderately sized real-life network. To reduce the computational load, Coupé & van der Gaag (2002) and van der Gaag et al. (2007) focus on obtaining the general form and general properties of sensitivity functions in one-way sensitivity analysis, where the parametric BNs only have one parameter. The latter additionally provides application insights in the context of robustness analysis and threshold decision making. Kwisthout & van der Gaag (2008) studied the theoretical complexity of tuning problems. Renooji (2014) studied the properties of the co-variation schemes for parametrization of BNs. Bolt & van der Gaag (2015, 2016, 2017) proposed *balanced* and *combined* tuning heuristics for parametrization that are interesting for future work.

More recently, Leonelli (2019) looked at pBNs with nonlinear sensitivity functions for single parameter pBNs. Ballester-Ripoll & Leonelli (2022) proposed a method to efficiently compute the derivatives of sensitivity functions (sensitivity values) and thus select the most relevant parameters to a query. They (a) limit to single parameter pBNs and (b) do not extend to synthesize the parameters.

Credal and constrained networks. Bolt et al. (2016) applied the BN sensitivity functions to improve inference on Credal networks (Cozman, 2000), BNs with imprecise rather than point probabilities. Some tool support for inference and sensitivity analysis on Credal networks is given by Tolo et al. (2018). The investigation of Credal network techniques and tools is left for future work; in particular, the connection to recent works on verifying uncertain Markov chains (Badings et al., 2021) would be of interest. Beaumont & Huth (2017) investigated constrained BNs. These networks are similarly defined as the parametric BNs in this paper. We are unaware of specific analysis algorithms for constrained BNs.

Structure learning. The connection to structure learning and parameter learning (Heckerman, Geiger, & Chickering, 1995; Heckerman, 2008; Grüttemeier & Komusiewicz, 2022) is of interest. Structure learning techniques e.g., Friedman & Koller (2003) and Cussens et al. (2017) aim at learning the structure of the BN from data. We focused on fine-tuning the parameters of a BN, where the dependency structure as a DAG is known.

Poon & Domingos (2011) introduced sum-product networks (SPNs) that are trending probabilistic graphical models. SPNs are tractable for several problems including inference. Zhao et al. (2015) investigated the connection of BNs and SPNs: they proved that every SPN converts into a BN in linear time and space in terms of the network size. There are several studies e.g., (Gens & Domingos, 2013; Trapp et al., 2019) on learning the structure and parameters of SPNs. In the context of sensitivity analysis, Conaty et al. (2019) consider the robustness of the decisions: they consider hierarchical SPNs. The idea is to carry over the classification task to another model, when the current SPN is sensitive to small perturbations of the parameters.

10. Epilogue

This paper treated several analysis problems for Bayesian networks that contain symbolic expressions (over a fixed set of parameters) in the conditional probability tables. We have shown how these analysis problems can be tackled in a fully automated manner using state-of-the-art parameter synthesis techniques from the field of probabilistic verification. This is enabled by a simple translation from Bayesian networks into Markov chains. A prototypical tool-chain that is publicly available[19] supports all reported capabilities.

Classical Bayesian networks. We started off by considering the classical setting without parameters and experimentally compared explicit-state as well as decision diagram techniques. In general, symbolic inference using PSDDs, a decision diagram tailored to inference, outperforms analysing the Markov chains using probabilistic model checking using MTBDDs (aka: algebraic decision diagrams). When MTBDDs are combined with model reduction techniques such as bisimulation minimization, model checking can outperform PSDD analysis. In addition, the use of temporal logic provides a flexible way to specify objectives including first-order combinations of random variables in the network.

Parametric Bayesian networks. For the parametric setting, we focused on three different objectives: (a) computing symbolic expressions (in terms of the model parameters) for inference, (b) feasibility (find a parameter valuation satisfying a threshold on the inference query), (c) partitioning the parameter space into accepting and rejecting regions. Our experimental results show that ideas and techniques for parameter synthesis in Markov models yield competitive methods for parametric Bayesian networks. We consider two aspects of importance. On the one hand, parameter space partitioning seems a new and potentially useful technique for analysing parametric Bayesian networks. On the other hand, the use of parameter synthesis techniques is (in theory) neither restricted to the number of parameters occurring in the network nor to where (i.e., in which conditional probability tables) these parameters occur. In particular, non-trivial parameter dependencies where parameters occur in multiple probability tables can be treated naturally. Our experiments indicate that our parameter synthesis techniques have the potential to scale rather well: symbolic functions for large networks with hundreds of parameters can be computed in a few minutes, feasibility for involved objectives and networks with hundreds of parameters is possible in a matter of a few hundred seconds, and parameter partitioning for up to about ten parameters works well.

Future work. Several directions are of interest. This includes investigating specific parameter synthesis algorithms for specific graph structures in networks, considering other classes such as dynamic (Palaniappan & Thiagarajan, 2012) and recursive Bayesian networks (Jaeger, 2001; Williamson & Gabbay, 2005). It would also be of interest to investigate checking the monotonicity (Van der Gaag, Bodlaender, & Feelders, 2004) of parameters at the network level can accelerate synthesis; first studies for parametric Markov chains show promising results (Spel, Junges, & Katoen, 2019). Recent fragmentation techniques to compute solution functions could also be of interest for Bayesian networks (Fang et al., 2021).

19. `https://github.com/baharSlmn/storm-bn`

Acknowledgments

This research is funded by the ERC AdG Project FRAPPANT (Grant Nr. 787914). The authors thank Robin Drahovsky, Caroline Jabs, and Hans Vrapi for their implementation efforts. We also kindly thank Tim Quatmann, Sebastian Junges, Linus Heck, and Arthur Choi for their support and discussions.

A. Inference Query Correspondence

Here, we provide the proofs for Proposition 1 and Proposition 2. The results are formalized for non-parametric MCs of BNs and carry over to pMCs and pBNs in a straightforward manner.

Preliminaries and notations. Let $\mathcal{M} = (S, s_0, P)$ be a MC, $\mathit{Paths}(\mathcal{M})$ be the set of paths in $\mathcal{M}$ that start in the initial state s_0, and $\mathit{Paths}^*(\mathcal{M})$ the set of their finite prefixes. For the path $\pi = s_0 \dots s_n \in \mathit{Paths}^*(\mathcal{M})$, let $P(\pi) = \prod_{i=0}^{n-1} P(s_i, s_{i+1})$. For infinite paths, a probability measure Pr is defined using a standard cylinder set construction (Baier & Katoen, 2008, Ch. 10). Let $\mathit{pre}(s)$ be the direct predecessors of s and $\mathit{pre}^*(s)$ be its reflexive and transitive closure. Let $G \subseteq S$ be the set of goal states in $\mathcal{M}$ and $\Diamond G$ denote the set of paths in $\mathcal{M}$ that eventually reach G, i.e., $\Diamond G = \{\pi \in \mathit{Paths}(\mathcal{M}) \mid \exists i \in \mathbb{N} \,.\, \pi(i) \in G\}$. Let $\mathit{Paths}(\mathcal{M}, \Diamond G) = \mathit{Paths}(\mathcal{M}) \cap (S \setminus G)^* G$, and

$$\Pr_{\mathcal{M}}(\Diamond G) \;=\; \sum_{s_0 \dots s_n \in \mathit{Paths}(\mathcal{M}, \Diamond G)} P(s_0 \dots s_n). \tag{7}$$

For each MC, $\Pr(\Diamond G)$ is measurable. Let variable $p_s = \Pr_{\mathcal{M}}(s \models \Diamond G)$ denote the probability to reach G from state s. It follows that

- if G is not reachable from s, then $p_s = 0$,
- if $s \in G$, then $p_s = 1$, and
- For any state $s \in \mathit{pre}^*(G) \setminus G$,

$$p_s = \underbrace{\sum_{t \in S \setminus G} P(s,t) \cdot p_t}_{\text{reaching } G \text{ via state } t} + \underbrace{\sum_{u \in G} P(s,u)}_{\text{reaching } G \text{ in one step}} \quad . \tag{8}$$

This yields a system of linear equations with unique solution $\Pr_{\mathcal{M}}(\Diamond G) = p_{s_0}$.

Let $\Box G$ denote the set of paths in $\mathcal{M}$ that always satisfy G, i.e.,

$$\Box G \;\equiv\; \neg \Diamond \neg G. \text{ Thus} \tag{9}$$

$$\Pr(\Box G) \;=\; 1 - \Pr(\Diamond \neg G). \tag{10}$$

Evidence-agnostic translation. To prove Proposition 1, we start off by formalizing the definition of the evidence-agnostic MC.

Let $\varrho = (v_1, \ldots, v_m)$ be a topological order over the variables in V with respect to the DAG $\mathcal{G}$ and $\varrho(v)$ denote the index of variable v in the order. Let $E = E_1 \wedge \ldots \wedge E_k$ be the evidence, where E_i is in the form $v_{E_i} = d_{E_i}$ with $v_{E_i} \in V$ and

$$v_{E_1} <_\varrho v_{E_2} <_\varrho \ldots <_\varrho v_{E_k} \quad \text{i.e.,} \quad \varrho(v_{E_1}) < \varrho(v_{E_2}) < \ldots < \varrho(v_{E_k}).$$

In the sequel, we often denote E_k as E_{last}.

The hypothesis H can be any arbitrary logical formula over the atomic propositions $v_i = d_i$. W.l.o.g, let $H = H_1 \wedge \ldots \wedge H_l$ be the hypothesis, where H_i is in the form $v_{H_i} = d_{H_i}$ and $\varrho(v_{H_i}) < \varrho(v_{H_{i+1}})$. For the logical formula A, let *vars*(A) denote the set of variables that appear in A. Let $*$ denote the *don't care* value. For the variable $v_i \in V$, let $D^*_{v_i} = D_{v_i} \cup \{*\}$. A *variable state* is a function $s\colon D^*_{v_1} \times \ldots \times D^*_{v_m}$ that maps each variable $v_i \in V$ either to some d_i or to the *don't care* value. Intuitively speaking, $v_i = *$ if the value of variable $v \in V$ is either not yet determined or not needed any more. Let *States*(V) denote the set of all variable states. Let $s \in States(V)$.

- We write $s \models (v_i = d_i)$ iff $s(v_i) = d_i$ with $d_i \in D_{v_i}$.
- We have $s \models (v_i = s(v_i) \wedge \ldots \wedge v_m = s(v_m))$.
- For $V' \subseteq V$ and $s' \in Eval(V')$, we write

$$s \models s' \text{ iff } \bigwedge_{v \in V'} s(v_i) = s'(v_i).$$

- For the logical formula α over the atomic propositions $(v_i = d_i)$, we write

$$s \models \alpha \text{ iff } (v_i = s(v_i) \wedge \ldots \wedge v_m = s(v_m)) \implies \alpha.$$

Given the pBN $\mathcal{B} = (V, W, X, \Theta)$, the topological order ϱ on V, and the state $s \in States(V)$, let $s[v_i{=}d_i] = s^*$, where

- $s^* \in States(V)$,
- $s^*(v_i) = d_i$,
- $s^*(v) = s(v)$ for $v \neq v_i$ satisfying

$$\varrho(v) < \varrho(v_i) \text{ and } \exists c \in children(v) \,.\, \varrho(c) > \varrho(v_i), \text{ and}$$

- $s^*(v) = *$ for all other variables.

Definition 4. Evidence-agnostic pMC of pBN Let $\mathcal{B} = (V, W, X, \Theta)$ be a pBN and $\varrho = (v_1, \ldots, v_m)$ be a topological order over V. The *pMC* of $\mathcal{B}$ is $\mathcal{M}^\varrho_\mathcal{B} = (S, s_0, X, P)$, where:

- $s_0 = (v_1 = *, \dots, v_m = *)$ is the initial state,
- $S \subseteq \mathit{States}(V)$ is the set of states defined as follows:
  - $S = \bigcup\limits_{0 \le i \le m} S_i$
  - $S_0 = \{s_0\}$
  - $S_{i+1} = \{s_i[v_{i+1}{=}d_{i+1}] \mid \text{for } s_i \in S_i \text{ and } d_{i+1} \in D_{v_{i+1}}\}$ for $0 \le i < m$, and
- $P\colon S \times S \to [0,1]$ is the transition probability functions defined by the following rules.

1. Let $0 \le i < m$, $\overline{par} \in \mathit{Eval}(\mathit{parents}(v_{i+1}))$, and $s_{i+1} = s_i[v_{i+1}{=}d_{i+1}] \in S_{i+1}$. For each $s_i \in S_i$ and $d_{i+1} \in D_{v_{i+1}}$,

$$P(s_i, s_{i+1}) = \Theta(\overline{par})(d_{i+1}) \text{ iff } s_i \models \overline{par}.$$

2. $P(s_m, s_m) = 1$ for each $s_m \in S_m$.
3. $P(s_i, s_j) = 0$, otherwise.

Definition 4 ensures that:

$$\forall 0 \le i < m, \forall s_i \in S_i \cdot \exists! \, \overline{par} \in \mathit{Eval}(\mathit{parents}(v_{i+1})) \text{ such that } s_i \models \overline{par}.$$

Query correspondence for the evidence-agnostic MC. We prove Proposition 1 by indicating the strong correspondence between the paths of the evidence-agnostic MC of $\mathcal{B}$ and the tree-like MC (Salmani & Katoen, 2020) of $\mathcal{B}$. Let $\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}} = (\boldsymbol{S}, \boldsymbol{s_0}, \boldsymbol{P})$ be the tree-like MC of $\mathcal{B}$ given the topological order ϱ. The construction of $\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}}$ (Salmani & Katoen, 2020) is similar to that of $\mathcal{M}_{\mathcal{B}}^{\varrho}$ in Def. 4 and only deviates in the following. Let $\boldsymbol{s_{i-1}}$ be a state at the $i{-}1$'th level of $\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}}$. The successor states of $\boldsymbol{s_{i-1}}$ are obtained for each $d_i \in D_{v_i}$ by $\boldsymbol{s_{i-1}}[v_i = d_i] = \boldsymbol{s}$, where

- $\boldsymbol{s}(v_i) = d_i$ and
- $\boldsymbol{s}(v) = \boldsymbol{s_{i-1}}(v)$ for $v \neq v_i$.

The transition probabilities are determined analogously to that of $\mathcal{M}_{\mathcal{B}}^{\varrho}$. This tree-like MC is thus obtained by considering all possible valuations of the variables in BN $\mathcal{B}$ adhering to the topological order ϱ. Inference on the BN $\mathcal{B}$ can be reduced to computing reachability probabilities on the tree-like MC $\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}}$, as

$$\Pr_{\mathcal{B}}(E) \;=\; \Pr_{\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}}}(\Diamond\, E) \quad \text{and} \quad \Pr_{\mathcal{B}}(H \wedge E) \;=\; \Pr_{\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}}}(\Diamond\, (H \wedge E)). \tag{11}$$

This is formally shown in (Salmani & Katoen, 2020).

Let $A \subseteq V$ be a subset of BN variables and $\eta \in \mathit{Eval}(A)$ be an evaluation of variables in A. Let $\mathit{Paths}(\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}}, \Diamond\, \eta) = \boldsymbol{s_0 s_1} \ldots \in \mathit{Paths}(\boldsymbol{\mathcal{M}_{\mathcal{B}}^{\varrho}})$ such that $\exists \boldsymbol{s_k}$ with $\boldsymbol{s_k}(v){=}\eta(v)$ for all $v{\in}A$. Let $\mathit{Paths}(\mathcal{M}_{\mathcal{B}}^{\varrho}, \Box\, \eta^*) = s_0 s_1 \ldots \in \mathit{Paths}(\mathcal{M}_{\mathcal{B}}^{\varrho})$ such that for each s_i and each $v{\in}A, s_i(v){=}*$ or $s_i(v){=}\eta(v)$.

Lemma A.1 *Path $\pi = s_0 s_1 \ldots \in Paths(\mathcal{M}^{\varrho}_{\mathcal{B}}, \Box\, \eta^*)$ if and only if there is the corresponding path $\boldsymbol{\pi} = \boldsymbol{s_0 s_1} \ldots \in Paths(\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}, \Diamond \eta)$ such that for each $i \in \mathbb{N}$,*

$$P(s_i, s_{i+1}) = \boldsymbol{P}(\boldsymbol{s_i}, \boldsymbol{s_{i+1}}).$$

Proof. We indicate the correspondence between the paths in one direction. The proof is analogous for the other direction. Let $\boldsymbol{\pi} = \boldsymbol{s_0 s_1} \ldots \in Paths(\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}, \Diamond \eta)$. By the definition of $\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}$,

- $\boldsymbol{s_0}$ is the initial state,
- for $1 \leq i \leq m$, $\boldsymbol{s_i} \in \boldsymbol{S_i}$ is state at the i'th level of $\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}$, and
- for each $\boldsymbol{s_i} \in \boldsymbol{S_i}$, $\exists!\, d_i \in D_{v_i}$ such that $\boldsymbol{s_i} \models (v_i = d_i)$[20].

Let $vals_{\boldsymbol{\pi}} = d_1 d_2 \cdots d_m$ be the sequence of the values for BN variables along the path $\boldsymbol{\pi}$, where $vals_{\boldsymbol{\pi}}(i) = d_i \in D_{v_i}$ iff $\boldsymbol{s_i} \models (v_i = d_i)$. The corresponding path $\pi \in Paths(\mathcal{M}^{\varrho}_{\mathcal{B}}, \Box \eta^*)$ is then obtained as follows.

Let $0 \leq i < m$ and $s_i \in S_i$ be a state at the i'th level of $\mathcal{M}^{\varrho}_{\mathcal{B}}$. Def. 4 ensures that

- for each $d_{i+1} \in D_{v_{i+1}}$, $\exists!\, s_{i+1} \in succ(s_i)$ such that $s_{i+1} \models (v_{i+1} = d_{i+1})$.

Given $vals_{\boldsymbol{\pi}}$, the path $\pi = s_0 s_1 \cdots \in Paths(\mathcal{M}^{\varrho}_{\mathcal{B}})$ is then obtained, where

- s_0 is the initial state of $\mathcal{M}^{\varrho}_{\mathcal{B}}$,
- for $0 \leq i < m$, s_{i+1} is the unique state in $succ(s_i)$ with
$$s_{i+1} \models (v_{i+1} = vals_{\boldsymbol{\pi}}(i{+}1)),$$
- and for $i \geq m$, $s_{i+1} = s_m$.

Note that for $0 \leq i < m$, $P(s_i, s_{i+1}) = \boldsymbol{P}(\boldsymbol{s_i}, \boldsymbol{s_{i+1}}) = \Theta(\overline{par})(vals_{\boldsymbol{\pi}}(i{+}1))$ for the parent valuation $\overline{par} \in Eval(parents(v_{i+1}))$ with $s_i \models \overline{par}$. For $i \geq m$, $P(s_i, s_{i+1}) = \boldsymbol{P}(\boldsymbol{s_i}, \boldsymbol{s_{i+1}}) = 1$.

It remains to show that $\pi \in Paths(\mathcal{M}^{\varrho}_{\mathcal{B}}, \Box \eta^*)$:

(a) Path $\boldsymbol{\pi} = \boldsymbol{s_0 s_1} \cdots \in Paths(\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}, \Diamond \eta)$. Thus, $\exists k$ such that $\boldsymbol{s_k}(v) = \eta(v)$ for all the variables $v \in A$.

(b) The BN variable v is not assigned to two distinct values in D_v along a given path in $Paths(\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}})$, i.e., for $d \in D_v$,

$$\boldsymbol{s_k}(v) = d \implies \boldsymbol{s_i}(v) = d \vee \boldsymbol{s_i}(v) = * \text{ for } i < k \text{ and } i > k.$$

(c) Let $\boldsymbol{s_i} \in \boldsymbol{S_i}$ be a state at the i'th level of $\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}$ and v_i be the i'th variable in the topological order. Then by the definition of $\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}$, $\boldsymbol{s_i}(v_i) \neq *$.

It follows by (a), (b), and (c) that $s_i(v_i) = vals_{\boldsymbol{\pi}}(i) = \eta(v_i)$ for $v_i \in A$. Moreover, similarly to argument (b), it holds that for $d_i \in D_{v_i}$,

$$s_i(v_i) = d_i \implies s_j(v_i) = d_i \vee s_j(v_i) = * \text{ for } j < i \text{ and } j > i.$$

20. Note that v_i is the i'th variable in the topological order ϱ.

It thus yields $\pi \in \mathit{Paths}(\mathcal{M}^{\varrho}_{\mathcal{B}}, \Box\eta^*)$. This ends the proof for Lemma A.1. $\boxtimes$

For the evidence $E = \bigwedge\limits_{i=1}^{k} (v_i = d_i)$, let $E^* = \bigwedge\limits_{i=1}^{k} (v_i = d_i \vee v_i = *)$.[21] The following equations are obtained by considering $\eta := E$ and $\eta := E \wedge H$ in Lemma A.1.

$$\Pr_{\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}}(\Diamond\, E) \;=\; \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box\, E^*) \quad \text{and} \quad \Pr_{\boldsymbol{\mathcal{M}^{\varrho}_{\mathcal{B}}}}(\Diamond\,(H \wedge E)) \;=\; \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box\,(H^* \wedge E^*)). \tag{12}$$

Proposition 1. For the pMC $\mathcal{M}^{\varrho}_{\mathcal{B}}$,

$$\Pr_{\mathcal{B}}(E) \;=\; 1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,\neg E) \quad \text{and} \quad \Pr_{\mathcal{B}}(H \mid E) \;=\; \frac{1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,(\neg H \vee \neg E))}{1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,\neg E)},$$

where the latter equality requires $\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,\neg E) < 1$.

Proof. It follows from equations (11) and (12) that

$$\Pr_{\mathcal{B}}(E) \;=\; \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box\, E^*) \quad \text{and} \; \Pr_{\mathcal{B}}(H \wedge E) \;=\; \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box\,(H^* \wedge E^*)). \tag{13}$$

Without loss of generality, assume that the BN variables in V are binary-valued with $D_{v_i} = \{d_i, \neg d_i\}$. Definition 4 ensures that for all $s \in S$, $s \models (v_i = d_i) \vee (v_i = \neg d_i) \vee (v_i = *)$. Let $\neg E = \bigvee_{i=1}^{k}(v_i = \neg d_i)$. It follows that

$$\neg(E^*) \;\equiv\; \neg E, \quad \text{thus}$$

$$\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box\, E^*) \;=\; 1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,\neg(E^*)) \;=\; 1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,\neg E). \tag{14}$$

Since E allows the conjunction of multiple variables, equation (14) is analogously valid for $(E \wedge H)$, i.e.,

$$\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box\,(H^* \wedge E^*)) \;=\; 1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,(\neg H \vee \neg E)). \tag{15}$$

We can then derive:

$$\begin{aligned} & \Pr_{\mathcal{B}}(H \mid E) \\ = \;& \frac{\Pr_{\mathcal{B}}(H \wedge E)}{\Pr_{\mathcal{B}}(E)} \\ \overset{(13)}{=} \;& \frac{\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}\big(\Box\,(H^* \wedge E^*)\big)}{\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box E^*)} \\ \overset{(14) \text{ and } (15)}{=} \;& \frac{1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,(\neg H \vee \neg E))}{1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Diamond\,\neg E)}. \end{aligned}$$

This finalizes the proof for Proposition 1. $\boxtimes$

21. H^* is defined analogously.

Evidence-tailored translation. We now proceed towards the proof of Proposition 2 by first formalizing the two operations to construct the *evidence-tailored* MC of BN: *propagation* and *redirection.*

Definition 5. (Propagation) Let $\mathcal{M}_{\mathcal{B}}^{\varrho}$ be the evidence-agnostic MC of BN $\mathcal{B}$ for the topological order ϱ. Applying the propagation operation on $\mathcal{M}_{\mathcal{B}}^{\varrho}$ yields the MC $\mathcal{M}_{\mathcal{B}}^{\varrho\downarrow} = (S^{\downarrow}, s_0^{\downarrow}, P^{\downarrow})$, where

– $s_0^{\downarrow} = s_0$ is the initial state.

– $S^{\downarrow} = \bigcup_{i=0}^{m} S_i^{\downarrow}$ is the set of states defined as follows:

  – For i=0 and $i > \varrho(v_{E_{last}})$,
  $S_i^{\downarrow} = S_i$, where S_i is the set of states at level i of $\mathcal{M}_{\mathcal{B}}^{\varrho}$.

  – For $0 < i \leq \varrho_{E_{last}}$,
  $S_i^{\downarrow} = \{\underbrace{s_{i-1}^{\downarrow}[v_i{=}d_i]}_{\text{short } s^{\downarrow}} \mid \text{for } s_{i-1}^{\downarrow} \in S_{i-1}^{\downarrow} \text{ and } d_i \in D_{v_i}\},$ with

  - $s^{\downarrow}(v_i) = d_i$,
  - $s^{\downarrow}(v) = s_{i-1}^{\downarrow}(v)$ for $v \neq v_i$ iff either

    (I) $\varrho(v) < \varrho(v_i)$ and $\exists c \in \mathit{children}(v) \,.\, \varrho(c) > \varrho(v_i)$ or

    (II) $v \notin \mathit{vars}(E)$,

  - and $s^{\downarrow}(v) = *$, otherwise.

– $P^{\downarrow}$ is defined analogously to P.

The MC $\mathcal{M}_{\mathcal{B}}^{\varrho\downarrow}$ differs from $\mathcal{M}_{\mathcal{B}}^{\varrho}$ for $0 < i \leq \varrho(v_{E_{last}})$ as imposed by constraint (II).

Lemma A.2 *The propagation operation ensures that*

$$\Pr_{\mathcal{B}}(H|E) \;=\; \frac{\Pr_{\mathcal{M}_{\mathcal{B}}^{\varrho\downarrow}}(\Box(H^* \wedge E^*))}{\Pr_{\mathcal{M}_{\mathcal{B}}^{\varrho\downarrow}}(\Box E^*)}. \tag{16}$$

Proof [22]. Definition 5 ensures a strong correspondence between the paths in $\mathcal{M}_{\mathcal{B}}^{\varrho}$ and $\mathcal{M}_{\mathcal{B}}^{\varrho\downarrow}$: $\pi^{\downarrow} = s_0^{\downarrow} s_1^{\downarrow} \ldots \in \mathit{Paths}(\mathcal{M}_{\mathcal{B}}^{\varrho\downarrow}, \Diamond E)$ if and only if there is the unique path $\pi = s_0 s_1 \ldots \in \mathit{Paths}(\mathcal{M}_{\mathcal{B}}^{\varrho}, \Box E^*)$ such that for each $i \in \mathbb{N}$,

$$P^{\downarrow}(s_i^{\downarrow}, s_{i+1}^{\downarrow}) = P(s_i, s_{i+1}).$$

22. The proof is analogous to Lemma A.1.

It follows that

$$\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box E^*) \;=\; \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*) \tag{17}$$

and analogously for $H \wedge E$,

$$\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B}}}(\Box(H^* \wedge E^*)) \;=\; \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box(H^* \wedge E^*)). \tag{18}$$

Equation (16) derives from the equations (17) and (18), and Proposition 1. ⊠

We now proceed by formalizing the redirection operation for an arbitrary MC. Let $\mathcal{M} = (S, s_0, P)$ be a MC and $AP = \{a_i \mid i \in \mathbb{N}\}$ be a set of atomic propositions. Let $L : S \to 2^{AP}$ be a labelling function that maps each state of $\mathcal{M}$ to a finite subset of atomic propositions. Let $\phi = \bigwedge_{i=1}^{k} a_i$ and $S_{\neg\phi}$ denote the set of states that violate ϕ, i.e.,

$$S_{\neg\phi} = \{s \in S \mid s \models \neg a_i \text{ for some } 1 \le i \le k\}.$$

The *redirection operation* reroutes the direct transitions to the states in $S_{\neg\phi}$ to the initial state s_0 and deletes the states in $S_{\neg\phi}$.

Definition 6. (Redirection) Applying the redirection operation to the MC $\mathcal{M}$ with respect to ϕ yields the MC $(\mathcal{M})_\phi = (S \setminus S_{\neg\phi}, s_0, P_\phi)$, where for $s \in S \setminus S_{\neg\phi}$

$$\begin{cases} P_\phi(s, s_0) = P(s, s_0) \; + \displaystyle\sum_{s_\neg \in S_{\neg\phi}} P(s, s_\neg) & \text{and} \\ P_\phi(s, s') = P(s, s') & \text{for } s' \in S \setminus S_{\neg\phi} \text{ with } s' \neq s_0. \end{cases}$$

Let G be a set of target states in $(\mathcal{M})_\phi$. Recall that variable p_s denotes the probability to reach G from state s. Let $succ(s)$ denote the set of direct successors of s in the MC $\mathcal{M}$. It follows by equation (8) and Def. 6 that for the state $s \in S \setminus S_{\neg\phi}$,

$$p_s \;=\; \sum_{s' \in succ(s) \setminus S_{\neg\phi}} P(s, s') \cdot p_{s'} \;+\; \underbrace{\sum_{s_\neg \in succ(s) \cap S_{\neg\phi}} P(s, s_\neg) \cdot p_{s_0}}_{\text{redirection to the initial state}}. \tag{19}$$

Query correspondence for the evidence-tailored MC. The evidence-tailored MC $\mathcal{M}^{\varrho}_{\mathcal{B},E} = (S^{\downarrow} \setminus S^{\downarrow}_{\neg E}, s^{\downarrow}_0, P_E)$ is obtained by applying the redirection operation to the MC $\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}$ with respect to $\phi = E$, i.e., $\mathcal{M}^{\varrho}_{\mathcal{B},E} = (\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}})_E$. Let $S_E = \bigcup_{i=\varrho(v_{E_{last}})}^{m} S^{\downarrow}_i$ be the set of states in $\mathcal{M}^{\varrho}_{\mathcal{B},E}$ that only occurs at the level of the last evidence node and afterwards, i.e., the set of states that occur after the last redirection edge, see Fig. 8.

Lemma A.3 *For $G \subseteq S_E$, it holds that*

$$\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}(\Diamond G) \;=\; \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Diamond G \wedge \Box E^*)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*)}.$$

Proof. We prove the above lemma by rewriting equation (19) for $\mathcal{M}^{\varrho}_{B,E}$. First, let $G \subseteq S^{\downarrow}$ be arbitrary. Recall that $S^{\downarrow}_i$ denotes the set of states at the i-th level of $\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}$.[23]

- For each $s^{\downarrow}_i \in S^{\downarrow}_i \cap pre^*(G)$,

$$p_{s^{\downarrow}_i} = \sum_{s^{\downarrow}_{i+1} \in S^{\downarrow}_{i+1} \setminus S^{\downarrow}_{\neg E}} P^{\downarrow}(s^{\downarrow}_i, s^{\downarrow}_{i+1}) \cdot p_{s^{\downarrow}_{i+1}} \; + \sum_{t^{\downarrow}_{i+1} \in S^{\downarrow}_{i+1} \cap S^{\downarrow}_{\neg E}} P^{\downarrow}(s^{\downarrow}_i, t^{\downarrow}_{i+1}) \cdot p_{s^{\downarrow}_0},$$

- for each $s^{\downarrow}_i \notin pre^*(G)$, $p_{s^{\downarrow}_i} = 0$,
- and for each $s^{\downarrow}_i \in G$, $p_{s^{\downarrow}_i} = 1$.

Let $\neg E_{1\ldots j} = \bigcup_{i=1}^{\varrho(v_{E_j})} S^{\downarrow}_i \cap S^{\downarrow}_{\neg E}$ specify the states that violate E_1 up to E_j. The above equation system yields the following.

For $G \subseteq \bigcup_{i=\varrho(v_{E_j})}^{m} S^{\downarrow}_i$,

$$p_{s^{\downarrow}_0} = \sum_{\pi \in Paths\left(\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}, \lozenge G\right) \setminus Paths\left(\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}, \lozenge\neg E_{1\ldots j}\right)} \Pr(\pi) \quad + \quad p_{s^{\downarrow}_0} \cdot \sum_{\pi' \in Paths\left(\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}, \lozenge\neg E_{1\ldots j}\right)} \Pr(\pi')$$

and for $G \subseteq \bigcup_{i=\varrho(v_{E_{last}})}^{m} S^{\downarrow}_i$,

$$p_{s^{\downarrow}_0} = \sum_{\pi \in Paths\left(\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}, \lozenge G\right) \setminus Paths\left(\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}, \lozenge\neg E\right)} \Pr(\pi) \quad + \quad p_{s^{\downarrow}_0} \cdot \sum_{\pi' \in Paths\left(\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}, \lozenge\neg E\right)} \Pr(\pi') \,. \tag{20}$$

We have $\neg(\lozenge\neg E) \equiv \Box E^*$ (Equations (9) and (14)). Moreover, note that $A \setminus B = A \cap \neg B$. We can thus rewrite equation (20) as

$$\Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}(\lozenge G) \;=\; p_{s^{\downarrow}_0} \;=\; \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\lozenge G \wedge \Box E^*) \;+\; p_{s^{\downarrow}_0} \cdot \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\lozenge\neg E).$$

Thus:

$$p_{s^{\downarrow}_0} \;=\; \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}(\lozenge G) \;=\; \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\lozenge G \wedge \Box E^*)}{1 - \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\lozenge\neg E)}.$$

Note that $1 - \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\lozenge\neg E) = \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*)$. This ends the proof. ⊠

Proposition 2. For the evidence-tailored pMC $\mathcal{M}^{\varrho}_{\mathcal{B},E}$ of pBN $\mathcal{B}$, we have:

$$\Pr_{\mathcal{B}}(H \mid E) \;=\; 1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}}\left(\lozenge\left((\neg H_{before} \wedge E_{last}) \vee \neg H_{after}\right)\right).$$

23. Note that by Def. 5, for every $s^{\downarrow}_i \in S^{\downarrow}_i$, $succ(s^{\downarrow}_i) \subseteq S^{\downarrow}_{i+1}$.

Proof. We apply Lemma A.3 to the evidence-tailored MC $\mathcal{M}^{\varrho}_{\mathcal{B},E}$. Let $G = \big((\neg H_{\mathit{before}} \wedge E_{\mathit{last}}) \vee \neg H_{\mathit{after}})\big)$. Recall that S_E is the set of states at the levels i with $i \geq \varrho(v_{E_{\mathit{last}}})$. We have $G \subseteq S_E$: the term $\neg H_{\mathit{before}} \wedge E_{\mathit{last}}$ can first be satisfied at level i with $i \geq \varrho(v_{E_{\mathit{last}}})$. The term H_{after} can also first be satisfied at levels i with $i > \varrho(v_{E_{\mathit{last}}})$ by the definition of H_{after}.

It then follows by Lemma A.3 that

$$
\begin{aligned}
& 1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}} \big(\Diamond G\big) \\
= \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big) - \Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Diamond G \wedge \Box E^*\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
\overset{\Pr(a)-\Pr(a\wedge b)=\Pr(a\wedge\neg b)}{=} \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box E^* \wedge \neg\Diamond G\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
\overset{\neg\Diamond a\equiv\Box\neg a}{=} \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box E^* \wedge \Box\neg G\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
\overset{\Box\neg(a\vee b)\equiv\Box\neg a\wedge\Box\neg b}{=} \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box E^* \wedge \Box\neg\, (\neg H_{\mathit{before}} \wedge E_{\mathit{last}}) \wedge \neg(\neg H_{\mathit{after}})\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
\overset{\neg(\neg H)\equiv H^*}{=} \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box E^* \wedge \Box(H^*_{\mathit{before}} \vee \neg E_{\mathit{last}}) \wedge \Box H^*_{\mathit{after}}\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
\overset{\Box a\wedge\Box b\equiv\wedge\Box(a\wedge b)}{=} \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box(E^* \wedge (H^*_{\mathit{before}} \vee \neg E_{\mathit{last}})) \wedge \Box H^*_{\mathit{after}}\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
= \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box((E^* \wedge H^*_{\mathit{before}}) \vee (E^* \wedge \neg E_{\mathit{last}})) \wedge \Box H^*_{\mathit{after}}\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
\overset{E^*\wedge\neg E_{\mathit{last}}\equiv\texttt{false}}{=} \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box(E^* \wedge H^*_{\mathit{before}} \wedge H^*_{\mathit{after}})\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)} \\
\overset{H^*_{\mathit{before}}\wedge H^*_{\mathit{after}}\equiv H^*}{=} \; & \frac{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}} \big(\Box(E^* \wedge H^*)\big)}{\Pr_{\mathcal{M}^{\varrho\downarrow}_{\mathcal{B}}}(\Box E^*\big)}.
\end{aligned}
$$

It then follows by Lemma A.2 that

$$1 - \Pr_{\mathcal{M}^{\varrho}_{\mathcal{B},E}} \big(\Diamond\big((\neg H_{\mathit{before}} \wedge E_{\mathit{last}}) \vee \neg H_{\mathit{after}})\big) \;=\; \Pr_{\mathcal{B}}(H \mid E),$$

which finalizes the proof for Proposition 2. ⊠

References

Achim, T., Sabharwal, A., & Ermon, S. (2016). Beyond parity constraints: Fourier analysis of hash functions for inference. In *ICML*, Vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 2254–2262. JMLR.org.

Agrawal, D., Pote, Y., & Meel, K. S. (2021). Partition function estimation: A quantitative study. In *IJCAI*, pp. 4276–4285. ijcai.org.

Badings, T. S., Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., & Topcu, U. (2021). Scenario-based verification of uncertain parametric MDPs. *CoRR*, *abs/2112.13020*.

Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., & Somenzi, F. (1997). Algebraic decision diagrams and their applications. *Formal Methods Syst. Des.*, *10*(2/3), 171–206.

Baier, C., Clarke, E. M., Hartonas-Garmhausen, V., Kwiatkowska, M. Z., & Ryan, M. (1997). Symbolic model checking for probabilistic processes. In *ICALP*, Vol. 1256 of *Lecture Notes in Computer Science*, pp. 430–440. Springer.

Baier, C., de Alfaro, L., Forejt, V., & Kwiatkowska, M. (2018). Model checking probabilistic systems. In *Handbook of Model Checking*, pp. 963–999. Springer.

Baier, C., Hensel, C., Hutschenreiter, L., Junges, S., Katoen, J.-P., & Klein, J. (2020). Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination. *Inf. Comput.*, *272*, 104504.

Baier, C., & Katoen, J.-P. (2008). *Principles of Model Checking*. MIT Press.

Baier, C., Klein, J., Klüppelholz, S., & Märcker, S. (2014). Computing conditional probabilities in Markovian models efficiently. In *TACAS*, Vol. 8413 of *Lecture Notes in Computer Science*, pp. 515–530. Springer.

Ballester-Ripoll, R., & Leonelli, M. (2022). You only derive once (YODO): automatic differentiation for efficient sensitivity analysis in Bayesian networks. In *PGM*, Vol. 186 of *Proceedings of Machine Learning Research*, pp. 169–180. PMLR.

Bart, A., Koriche, F., Lagniez, J., & Marquis, P. (2016). An improved CNF encoding scheme for probabilistic inference. In *ECAI*, Vol. 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 613–621. IOS Press.

Bartocci, E., Kovács, L., & Stankovic, M. (2020). Analysis of Bayesian networks via prob-solvable loops. In *ICTAC*, Vol. 12545 of *Lecture Notes in Computer Science*, pp. 221–241. Springer.

Batz, K., Kaminski, B. L., Katoen, J.-P., & Matheja, C. (2018). How long, O Bayesian network, will I sample thee? - A program analysis perspective on expected sampling times. In *ESOP*, Vol. 10801 of *Lecture Notes in Computer Science*, pp. 186–213. Springer.

Beaumont, P., & Huth, M. (2017). Constrained Bayesian networks: Theory, optimization, and applications. *CoRR*, *abs/1705.05326*.

Bollig, B., & Wegener, I. (1996). Improving the variable ordering of OBDDs is NP-complete. *IEEE Trans. Computers*, *45*(9), 993–1002.

Bolt, J. H. (2016). Bayesian networks: a combined tuning heuristic. In *PGM*, Vol. 52 of *JMLR Workshop and Conference Proceedings*, pp. 37–49. JMLR.org.

Bolt, J. H., Bock, J. D., & Renooij, S. (2016). Exploiting Bayesian network sensitivity functions for inference in Credal networks. In *ECAI*, Vol. 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 646–654. IOS Press.

Bolt, J. H., & Van der Gaag, L. C. (2015). Balanced tuning of multi-dimensional Bayesian network classifiers. In *ECSQARU*, Vol. 9161 of *Lecture Notes in Computer Science*, pp. 210–220. Springer.

Bolt, J. H., & van der Gaag, L. C. (2017). Balanced sensitivity functions for tuning multi-dimensional Bayesian network classifiers. *Int. J. Approx. Reason.*, *80*, 361–376.

Bova, S. (2016). SDDs are exponentially more succinct than OBDDs. In *AAAI*, pp. 929–935. AAAI Press.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press.

Bryant, R. E. (2018). Binary decision diagrams. In *Handbook of Model Checking*, pp. 191–217. Springer.

Budde, C. E., Dehnert, C., Hahn, E. M., Hartmanns, A., Junges, S., & Turrini, A. (2017). JANI: quantitative model and tool interaction. In *TACAS (2)*, Vol. 10206 of *Lecture Notes in Computer Science*, pp. 151–168.

Castillo, E. F., Gutiérrez, J. M., & Hadi, A. S. (1995). Parametric structure of probabilities in Bayesian networks. In *ECSQARU*, Vol. 946 of *Lecture Notes in Computer Science*, pp. 89–98. Springer.

Castillo, E. F., Gutiérrez, J. M., & Hadi, A. S. (1996). Goal oriented symbolic propagation in Bayesian networks. In *AAAI/IAAI, Vol. 2*, pp. 1263–1268. AAAI Press / The MIT Press.

Castillo, E. F., Gutiérrez, J. M., & Hadi, A. S. (1997a). Sensitivity analysis in discrete Bayesian networks. *IEEE Trans. Syst. Man Cybern. Part A*, *27*(4), 412–423.

Castillo, E. F., Gutiérrez, J. M., Hadi, A. S., & Solares, C. (1997b). Symbolic propagation and sensitivity analysis in Gaussian Bayesian networks with application to damage assessment. *Artif. Intell. Eng.*, *11*(2), 173–181.

Chaki, S., & Gurfinkel, A. (2018). BDD-based symbolic model checking. In *Handbook of Model Checking*, pp. 219–245. Springer.

Chakraborty, S., Meel, K. S., Mistry, R., & Vardi, M. Y. (2016). Approximate probabilistic inference via word-level counting. In *AAAI*, pp. 3218–3224. AAAI Press.

Chakraborty, S., Meel, K. S., & Vardi, M. Y. (2013). A scalable approximate model counter. In *CP*, Vol. 8124 of *Lecture Notes in Computer Science*, pp. 200–216. Springer.

Chan, H., & Darwiche, A. (2002). When do numbers really matter?. *J. Artif. Intell. Res.*, *17*, 265–287.

Chan, H., & Darwiche, A. (2004). Sensitivity analysis in Bayesian networks: From single to multiple parameters. In *UAI*, pp. 67–75. AUAI Press.

Chan, H., & Darwiche, A. (2005). A distance measure for bounding probabilistic belief change. *Int. J. Approx. Reason.*, *38*(2), 149–174.

Chavira, M., & Darwiche, A. (2005). Compiling Bayesian networks with local structure. In *IJCAI*, pp. 1306–1312. Professional Book Center.

Chavira, M., & Darwiche, A. (2006). Encoding CNFs to empower component analysis. In *SAT*, Vol. 4121 of *Lecture Notes in Computer Science*, pp. 61–74. Springer.

Chavira, M., & Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artif. Intell.*, *172*(6-7), 772–799.

Chen, X., Niu, L., & Yuan, Y. (2013). Optimality conditions and a smoothing trust region Newton method for non-Lipschitz optimization. *SIAM J. Optim.*, *23*(3), 1528–1552.

Choi, A., Kisa, D., & Darwiche, A. (2013). Compiling probabilistic graphical models using sentential decision diagrams. In *ECSQARU*, Vol. 7958 of *Lecture Notes in Computer Science*, pp. 121–132. Springer.

Claret, G., Rajamani, S. K., Nori, A. V., Gordon, A. D., & Borgström, J. (2013). Bayesian inference using data flow analysis. In *ESEC/SIGSOFT FSE*, pp. 92–102. ACM.

Clarke, E. M., Grumberg, O., Jha, S., Lu, Y., & Veith, H. (2000). Counterexample-guided abstraction refinement. In *CAV*, Vol. 1855 of *Lecture Notes in Computer Science*, pp. 154–169. Springer.

Conaty, D., del Rincón, J. M., & de Campos, C. P. (2019). A hierarchy of sum-product networks using robustness. *Int. J. Approx. Reason.*, *113*, 245–255.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.*, *42*(2-3), 393–405.

Coupé, V. M. H., & Van der Gaag, L. C. (2002). Properties of sensitivity analysis of Bayesian belief networks. *Ann. Math. Artif. Intell.*, *36*(4), 323–356.

Coupé, V. M., & Van der Gaag, L. C. (1998). *Practicable sensitivity analysis of Bayesian belief networks*, Vol. 1998. Utrecht University: Information and Computing Sciences.

Coupé, V. M., Van der Gaag, L. C., & Habbema, J. D. F. (2000). Sensitivity analysis: an aid for belief-network quantification. *The Knowledge Engineering Review*, *15*(3), 215–232.

Cozman, F. G. (2000). Credal networks. *Artif. Intell.*, *120*(2), 199–233.

Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.-P., & Topcu, U. (2018). Synthesis in pMDPs: A tale of 1001 parameters. In *ATVA*, Vol. 11138 of *Lecture Notes in Computer Science*, pp. 160–176. Springer.

Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., & Topcu, U. (2022). Convex optimization for parameter synthesis in mdps. *IEEE Trans. Autom. Control.*, *67*(12), 6333–6348.

Cussens, J., Järvisalo, M., Korhonen, J. H., & Bartlett, M. (2017). Bayesian network structure learning with integer programming: Polytopes, facets and complexity. *J. Artif. Intell. Res.*, *58*, 185–229.

Dagum, P., & Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is np-hard. *Artif. Intell.*, *60*(1), 141–153.

Darwiche, A. (2002). A logical approach to factoring belief networks. In *KR*, pp. 409–420. Morgan Kaufmann.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks.* Cambridge University Press.

Darwiche, A. (2011). SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, pp. 819–826. IJCAI/AAAI.

Daws, C. (2004). Symbolic and parametric model checking of discrete-time Markov chains. In *ICTAC*, Vol. 3407 of *Lecture Notes in Computer Science*, pp. 280–294. Springer.

Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J.-P., & Ábrahám, E. (2015). Prophesy: A probabilistic parameter synthesis tool. In *CAV (1)*, Vol. 9206 of *Lecture Notes in Computer Science*, pp. 214–231. Springer.

Dehnert, C., Junges, S., Katoen, J.-P., & Volk, M. (2017). A storm is coming: A modern probabilistic model checker. In *CAV (2)*, Vol. 10427 of *Lecture Notes in Computer Science*, pp. 592–600. Springer.

Deininger, D., Dimitrova, R., & Majumdar, R. (2016). Symbolic model checking for factored probabilistic models. In *ATVA*, Vol. 9938 of *Lecture Notes in Computer Science*, pp. 444–460.

Dilkas, P., & Belle, V. (2021). Weighted model counting with conditional weights for Bayesian networks. In *UAI*, Vol. 161 of *Proceedings of Machine Learning Research*, pp. 386–396. AUAI Press.

Druzdzel, M. J., & Van der Gaag, L. C. (2000). Building probabilistic networks:" where do the numbers come from?". *IEEE Trans. Knowl. Data Eng.*, *12*(4), 481–486.

Ermon, S., Gomes, C. P., Sabharwal, A., & Selman, B. (2013). Embed and project: Discrete sampling with universal hashing. In *NIPS*, pp. 2085–2093.

Ermon, S., Gomes, C. P., Sabharwal, A., & Selman, B. (2014). Low-density parity constraints for hashing-based discrete integration. In *ICML*, Vol. 32 of *JMLR Workshop and Conference Proceedings*, pp. 271–279. JMLR.org.

Fang, X., Calinescu, R., Gerasimou, S., & Alhwikem, F. (2021). Fast parametric model checking through model fragmentation. In *ICSE*, pp. 835–846. IEEE.

Friedman, N., & Koller, D. (2003). Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Mach. Learn.*, *50*(1-2), 95–125.

Fu, C., Hahn, E. M., Li, Y., Schewe, S., Sun, M., Turrini, A., & Zhang, L. (2022). EPMC gets knowledge in multi-agent systems. In *VMCAI*, Vol. 13182 of *Lecture Notes in Computer Science*, pp. 93–107. Springer.

Fujita, M., McGeer, P. C., & Yang, J. C. (1997). Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods Syst. Des.*, *10*(2/3), 149–169.

Gainer, P., Hahn, E. M., & Schewe, S. (2018). Accelerated model checking of parametric Markov chains. In *ATVA*, Vol. 11138 of *Lecture Notes in Computer Science*, pp. 300–316. Springer.

Gens, R., & Domingos, P. M. (2013). Learning the structure of sum-product networks. In *ICML (3)*, Vol. 28 of *JMLR Workshop and Conference Proceedings*, pp. 873–880. JMLR.org.

Grüttemeier, N., & Komusiewicz, C. (2022). Learning Bayesian networks under sparsity constraints: A parameterized complexity analysis. *J. Artif. Intell. Res.*, *74*, 1225–1267.

Hahn, E. M., Hermanns, H., & Zhang, L. (2011). Probabilistic reachability for parametric Markov models. *Int. J. Softw. Tools Technol. Transf.*, *13*(1), 3–19.

Han, Y. (2013). State elimination heuristics for short regular expressions. *Fundam. Informaticae*, *128*(4), 445–462.

Hansson, H., & Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects Comput.*, *6*(5), 512–535.

Hartmanns, A., & Hermanns, H. (2014). The modest toolset: An integrated environment for quantitative modelling and verification. In *TACAS*, Vol. 8413 of *Lecture Notes in Computer Science*, pp. 593–598. Springer.

Heck, L., Spel, J., Junges, S., Moerman, J., & Katoen, J.-P. (2022). Gradient-descent for randomized controllers under partial observability. In *VMCAI*, Vol. 13182 of *Lecture Notes in Computer Science*, pp. 127–150. Springer.

Heckerman, D. (2008). A tutorial on learning with Bayesian networks. In *Innovations in Bayesian Networks*, Vol. 156 of *Studies in Computational Intelligence*, pp. 33–82. Springer.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.*, *20*(3), 197–243.

Holtzen, S., Junges, S., Vazquez-Chanlatte, M., Millstein, T. D., Seshia, S. A., & Van den Broeck, G. (2021). Model checking finite-horizon Markov chains with probabilistic inference. In *CAV (2)*, Vol. 12760 of *Lecture Notes in Computer Science*, pp. 577–601. Springer.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2003). *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley.

Hutschenreiter, L., Baier, C., & Klein, J. (2017). Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination. In *GandALF*, Vol. 256 of *EPTCS*, pp. 16–30.

Jaeger, M. (2001). Complex probabilistic modeling with recursive relational Bayesian networks. *Ann. Math. Artif. Intell.*, *32*(1-4), 179–220.

Jensen, F. V. (1999). Gradient descent training of Bayesian networks. In *ESCQARU*, Vol. 1638 of *Lecture Notes in Computer Science*, pp. 190–200. Springer.

Jerrum, M., Valiant, L. G., & Vazirani, V. V. (1986). Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, *43*, 169–188.

Junges, S., Ábrahám, E., Hensel, C., Jansen, N., Katoen, J.-P., Quatmann, T., & Volk, M. (2019). Parameter synthesis for Markov models. *CoRR*, *abs/1903.07993.*

Junges, S., Katoen, J.-P., Pérez, G. A., & Winkler, T. (2021). The complexity of reachability in parametric Markov decision processes. *J. Comput. Syst. Sci.*, *119*, 183–210.

Katoen, J.-P. (2016). The probabilistic model checking landscape. In *LICS*, pp. 31–45. ACM.

Katoen, J.-P., Kemna, T., Zapreev, I. S., & Jansen, D. N. (2007). Bisimulation minimisation mostly speeds up probabilistic model checking. In *TACAS*, Vol. 4424 of *Lecture Notes in Computer Science*, pp. 87–101. Springer.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster).*

Kisa, D., Van den Broeck, G., Choi, A., & Darwiche, A. (2014). Probabilistic sentential decision diagrams. In *KR*. AAAI Press.

Kjærulff, U., & Van der Gaag, L. C. (2000). Making sensitivity analysis computationally efficient. In *UAI*, pp. 317–325. Morgan Kaufmann.

Kwiatkowska, M. Z., Norman, G., & Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, Vol. 6806 of *Lecture Notes in Computer Science*, pp. 585–591. Springer.

Kwisthout, J., & Van der Gaag, L. C. (2008). The computational complexity of sensitivity analysis and parameter tuning. In *UAI*, pp. 349–356. AUAI Press.

Lanotte, R., Maggiolo-Schettini, A., & Troina, A. (2007). Parametric probabilistic transition systems for system design and analysis. *Formal Aspects Comput.*, *19*(1), 93–109.

Laskey, K. B. (1995). Sensitivity analysis for probability assessments in Bayesian networks. *IEEE Trans. Syst. Man Cybern.*, *25*(6), 901–909.

Leonelli, M. (2019). Sensitivity analysis beyond linearity. *Int. J. Approx. Reason.*, *113*, 106–118.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Han, J. (2020). On the variance of the adaptive learning rate and beyond. In *ICLR*. OpenReview.net.

McClelland, J. L., Rumelhart, D. E., Group, P. R., et al. (1986). *Parallel Distributed Processing*, Vol. 2. MIT Press.

Palaniappan, S. K., & Thiagarajan, P. S. (2012). Dynamic Bayesian networks: A factored model of probabilistic dynamics. In *ATVA*, Vol. 7561 of *Lecture Notes in Computer Science*, pp. 17–25. Springer.

Pipatsrisawat, K., & Darwiche, A. (2008). New compilation languages based on structured decomposability. In *AAAI*, pp. 517–522. AAAI Press.

Pipatsrisawat, T., & Darwiche, A. (2010). A lower bound on the size of decomposable negation normal form. In *AAAI*. AAAI Press.

Poon, H., & Domingos, P. M. (2011). Sum-product networks: A new deep architecture. In *UAI*, pp. 337–346. AUAI Press.

Puterman, M. L. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Quatmann, T., Dehnert, C., Jansen, N., Junges, S., & Katoen, J.-P. (2016). Parameter synthesis for Markov models: Faster than ever. In *ATVA*, Vol. 9938 of *Lecture Notes in Computer Science*, pp. 50–67.

Renooij, S. (2014). Co-variation for sensitivity analysis in Bayesian networks: Properties, consequences and alternatives. *Int. J. Approx. Reason.*, *55*(4), 1022–1042.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, *abs/1609.04747*.

Salmani, B., & Katoen, J.-P. (2020). Bayesian inference by symbolic model checking. In *QEST*, Vol. 12289 of *Lecture Notes in Computer Science*, pp. 115–133. Springer.

Salmani, B., & Katoen, J.-P. (2021). Fine-tuning the odds in Bayesian networks. In *ECSQARU*, Vol. 12897 of *Lecture Notes in Computer Science*, pp. 268–283. Springer.

Sang, T., Beame, P., & Kautz, H. A. (2005). Performing Bayesian inference by weighted model counting. In *AAAI*, pp. 475–482. AAAI Press / The MIT Press.

Sanner, S., & McAllester, D. A. (2005). Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. In *IJCAI*, pp. 1384–1390. Professional Book Center.

Scutari, M. (2019). Bayesian network repository. `https://www.bnlearn.com`. Accessed: 2019.

Sharma, S., Roy, S., Soos, M., & Meel, K. S. (2019). GANAK: A scalable probabilistic exact model counter. In *IJCAI*, pp. 1169–1176. ijcai.org.

Shih, A., Choi, A., & Darwiche, A. (2018). Formal verification of Bayesian network classifiers. In *PGM*, Vol. 72 of *Proceedings of Machine Learning Research*, pp. 427–438. PMLR.

Spel, J., Junges, S., & Katoen, J.-P. (2019). Are parametric Markov chains monotonic?. In *ATVA*, Vol. 11781 of *Lecture Notes in Computer Science*, pp. 479–496. Springer.

Stankovic, M., Bartocci, E., & Kovács, L. (2022). Moment-based analysis of Bayesian network properties. *Theor. Comput. Sci.*, *903*, 113–133.

Sutskever, I., Martens, J., Dahl, G. E., & Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. In *ICML (3)*, Vol. 28 of *JMLR Workshop and Conference Proceedings*, pp. 1139–1147. JMLR.org.

Tolo, S., Patelli, E., & Beer, M. (2018). An open toolbox for the reduction, inference computation and sensitivity analysis of Credal networks. *Adv. Eng. Softw.*, *115*, 126–148.

Trapp, M., Peharz, R., Ge, H., Pernkopf, F., & Ghahramani, Z. (2019). Bayesian learning of Sum-Product Networks. In *NeurIPS*, pp. 6344–6355.

Valmari, A., & Franceschinis, G. (2010). Simple $O(m \log n)$ time Markov chain lumping. In *TACAS*, Vol. 6015 of *Lecture Notes in Computer Science*, pp. 38–52. Springer.

Van der Gaag, L. C., Bodlaender, H. L., & Feelders, A. J. (2004). Monotonicity in Bayesian networks. In *UAI*, pp. 569–576. AUAI Press.

Van der Gaag, L. C., & Renooij, S. (2001). Analysing sensitivity data from probabilistic networks. In *UAI*, pp. 530–537. Morgan Kaufmann.

Van der Gaag, L. C., Renooij, S., & Coupé, V. M. (2007). Sensitivity analysis of probabilistic networks. In *Advances in Probabilistic Graphical Models*, pp. 103–124. Springer.

Williamson, J., & Gabbay, D. (2005). Recursive causality in bayesian networks and self-fibring networks. *Laws and models in the sciences*, 173–221.

Wimmer, R., Jansen, N., Vorpahl, A., Ábrahám, E., Katoen, J.-P., & Becker, B. (2015). High-level counterexamples for probabilistic automata. *Log. Methods Comput. Sci.*, *11*(1).

Yuan, Y. (2015). Recent advances in trust region algorithms. *Math. Prog.*, *151*(1), 249–281.

Zhao, H., Melibari, M., & Poupart, P. (2015). On the relationship between Sum-Product Networks and Bayesian networks. In *ICML*, Vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 116–124. JMLR.org.