Anytime Approximation in Probabilistic Databases via Scaled Dissociations

Maarten Van den Heuvel
University of Antwerp

Peter Ivanov
Northeastern University

Wolfgang Gatterbauer
Northeastern University

Floris Geerts
University of Antwerp

Martin Theobald
University of Luxembourg

ABSTRACT

Speeding up probabilistic inference remains a key challenge in probabilistic databases (PDBs) and the related area of statistical relational learning (SRL). Since computing probabilities for query answers is #P-hard, even for fairly simple conjunctive queries, both the PDB and SRL communities have proposed a number of approximation techniques over the years. The two prevalent techniques are either (*i*) MCMC-style sampling or (*ii*) branch-and-bound (B&B) algorithms that iteratively improve model-based bounds using a combination of variable substitution and elimination.

We propose a new *anytime B&B approximation scheme* that encompasses all prior model-based approximation schemes proposed in the PDB and SRL literature. Our approach relies on the novel idea of "*scaled dissociation*" which can improve both the upper and lower bounds of existing model-based algorithms. We apply our approach to the well-studied problem of evaluating self-join-free conjunctive queries over tuple-independent PDBs, and show a consistent reduction in approximation error in our experiments on TPC-H, Yago3, and a synthetic benchmark setting.

ACM Reference Format:
Maarten Van den Heuvel, Peter Ivanov, Wolfgang Gatterbauer, Floris Geerts, and Martin Theobald. 2019. Anytime Approximation in Probabilistic Databases via Scaled Dissociations. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3299869.3319900

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3319900

U	X	p
u_1	a	0.4
u_2	f	0.6

R	X	Y	p
r_1	a	b	0.3
r_2	a	e	0.4
r_3	f	e	0.6

S	Y	Z	p
s_1	b	c	0.8
s_2	b	d	0.4
s_3	e	d	0.3

T	Z	p
t_1	c	0.2
t_2	d	0.9

Figure 1: Example 1: PDB (D, p).

1 INTRODUCTION

Probabilistic databases (PDBs) [46] address the pressing need of managing large amounts of uncertain data directly inside a relational database system. Popular examples of large, uncertain datasets include those obtained from information extraction (e.g., Probase [49], DeepDive [45], or Google's Knowledge Vault [13]), sensors (e.g., RFID data [29]), and many others. Millions to billions of uncertain tuples reside in these datasets as the result of automatic extraction, integration, or transformation from various data sources. Apart from just managing uncertain data, query processing over uncertain data remains a major challenge in PDBs and the broader field of statistical relational learning (SRL) [36]. Query processing in PDBs is based on the possible worlds semantics [46] and is closely related to the problem of weighted model counting [24] over Boolean expressions in SRL.

Within the PDB context, the seminal dichotomy theorem by Dalvi and Suciu [8, 9] states that the query-answering problem for unions of conjunctive queries (UCQs) over a tuple-independent PDB (TI-PDB) has either PTIME data complexity or is #P-hard. A main result based on the theorem is that *self-join-free* (sj-free) CQs with PTIME data complexity can be characterized concisely as "hierarchical" queries [46]. These can be processed by an "extensional" query plan, in which the computation of the answer tuples' probabilities is coupled with the standard relational operations. For "non-hierarchical" queries, however, computing the exact marginal probabilities of the query answers is intractable, in general. Consequently, these queries need to be processed by an "intensional" query evaluation, based on *Boolean lineage expressions* [3, 18, 46], by which the marginal probabilities of the query answers can then be approximated.

Example 1 (Running example). *Consider the example PDB, denoted as (D, p), from Fig. 1. The following sj-free full*

CQ $Q_1(X,Y,Z) :- U(X), R(X,Y), S(Y,Z), T(Z)$ *is hierarchical and can be evaluated in PTIME. The criterion is that the existential variables of a query form a "hierarchy" (here all 3 variables are head variables and there are no existential variables). That is, all lineage expressions of query answers produced by* Q_1 *are guaranteed to be free of repeated occurrences of tuple identifiers [32]. The lineage expression for the query answer* (a,b,c) *to* Q_1 *over* D *is* $\varphi_{Q_1,D}(a,b,c) = u_1 r_1 s_1 t_1$ *(where we abbreviate conjunctions of the form* $x \wedge y$ *by writing* $x\,y$*). Assuming tuple independence, the marginal probability of this answer is computed as* $\mathbb{P}[\varphi_{Q_1,D}(a,b,c)] = 0.4 \cdot 0.3 \cdot 0.8 \cdot 0.2 \approx 0.019$.

However, the Boolean version of this query, $Q_2 :- U(X), R(X,Y), S(Y,Z), T(Z)$*, is not hierarchical, since the sets of relations in which the non-head variables* X*,* Y*, and* Z *occur do not form a hierarchy. Using the query plan* $U \bowtie (R \bowtie (S \bowtie T))$*, a possible factorization of the lineage expression for* Q_2 *over* D *is*

$$\varphi_{Q_2,D} = u_1(r_1(s_1t_1 \vee s_2t_2) \vee r_2s_3t_2) \vee u_2r_3s_3t_2.$$

In general, calculating this probability is intractable and needs to be approximated. Here, this resolves to $\mathbb{P}[\varphi_{Q_2,D}] \approx \mathbf{0.170}$. ◇

Our Focus. We focus on these intractable cases and devise an *anytime approximation framework to approximate the probability of a monotone Boolean lineage* with guaranteed upper and lower bounds. An "anytime algorithm" is an algorithm that can return an approximate solution to a problem for any allocation of time, and finds better and better solutions the longer it keeps running [5, 50]. We develop our approach for non-hierarchical sj-free CQs and their associated lineages in the widely studied tuple-independent (TI) data model. Since the hardness results already apply to this simple model, it is apt to first consider TI-PDBs before moving to more complicated models that take correlations into account (e.g., [43]). Moreover, it was shown that correlations (e.g., as introduced by a Markov Logic Network) can be rewritten into a query over a tuple independent probabilistic database [22, 26]. In this way, TI-PDBs are a basic building block for a wide range of applications. Also notice that calculating the marginal probability of a Boolean lineage expression is closely related to the *weighted model counting* (WMC) problem [42], and it is well known that probabilistic inference in probabilistic graphical models can be reduced to WMC [7]. Thus our approach is also of interest to the general probabilistic reasoning and SRL communities.

State-of-the-Art. A number of methods for approximating the marginal probabilities of query answers have been developed (see Sect. 8 as well as [12] for a recent overview). The two prevalent techniques are either based on (*i*) *MCMC-style sampling* [6, 8, 37] or on (*ii*) *branch-and-bound* (B&B) methods. The latter iteratively compute *model-based bounds* [16, 17, 35] (see Example 2 below) over a Boolean lineage expression by using a combination of variable substitution and elimination. Both approaches allow *anytime approximations*: the more time the approaches are allowed to run, the better the bounds become. While MCMC-based approaches are the more common approach for approximate probabilistic inference, B&B approaches have been shown to work faster and better for monotone queries, such as UCQs [16]. In this context, our *anytime approximation framework* relies on the following three ingredients:

(1) Scaled Dissociations. Recent work [19] suggests so-called "dissociation"-based bounds for monotone Boolean formulas. These bounds were shown to generalize model-based bounds, but have so far only been applied as one-shot approximations at the query level [19–21]. In addition, empirical results of [19–21] showed that, while the upper bounds appear to work well (they are provably better than all model-based bounds), the lower "symmetric" bounds obtained by plan-level dissociations usually are poor approximations. In fact, they can often be far worse than the best model-based lower bounds. One of our key contributions is the introduction of "*scaled dissociations*", i.e., dissociation-based lower bounds that are guaranteed to be – at least theoretically – the best lower oblivious bounds and to thus always dominate model-based lower bounds.

Example 2 (Running example cont.). *Consider* $\varphi_{Q_2,D}$ *from Example 1. Its lineage-level dissociation yields*

$$\varphi'_{Q_2,D} = u_1(r_1(s_1t_1 \vee s_2t'_2) \vee r_2s'_3t''_2) \vee u_2r_3s''_3t'''_2$$

where all repeated occurrences of s_3 *and* t_2 *are replaced by fresh variables (indicated by apostrophes). In [19], it was shown that the best "oblivious" upper bound (UB) for* $\mathbb{P}[\varphi_{Q_2,D}] \approx \mathbf{0.170}$ *is obtained by assigning* $p_U(t'_2) = p_U(t''_2) = p_U(t'''_2) = p(t_2) = 0.9$*,* $p_U(s'_3) = p_U(s''_3) = p(s_3) = 0.3$ *and* $p_U(x) = p(x)$ *for all other tuple identifiers* x *in* φ'*, thus resulting in* $\mathbb{P}_U[\varphi'] \approx \mathbf{0.180}$*. By contrast, for lower bounding* $\mathbb{P}[\varphi_{Q_2,D}]$*, we obtain a* continuum *of oblivious lower bounds (LBs) by assigning probabilities* p_L *to* $\{t'_2, t''_2, t'''_2, s'_3, s''_3\}$ *s.t.* $\bar{p}_L(t'_2)\cdot\bar{p}_L(t''_2)\cdot\bar{p}_L(t'''_2) = \bar{p}(t_2) = 0.1$ *and* $\bar{p}_L(s'_3) \cdot \bar{p}_L(s''_3) = \bar{p}(s_3) = 0.7$ *(where, here and later,* $\bar{p} := 1-p$*), and* $p_L(x) = p(x)$ *for all other identifiers.*

One special case are the model-based *(MB) bounds of [16, 17, 34], in which a single occurrence of an identifier obtains all of the probability mass. In this example, the best MB LB* $\mathbb{P}_L[\varphi'] \approx \mathbf{0.110}$ *is obtained by* $p_L([t'_2, t''_2, t'''_2]) = [0, 0, 0.9]$ *and* $p_L([s'_3, s''_3]) = [0, 0.3]$*. An exhaustive enumeration of all 6 possible model-based bounds is needed to find the best one.*

Alternatively, one may also generate a symmetric dissociation *(SD) LB [20, 21] in which all occurrences get the same share, i.e.,* $p_L(t'_2) = p_L(t''_2) = p_L(t'''_2) = 1 - \bar{p}(t_2)^{1/3} \approx 0.535$ *and* $p_L(s'_3) = p_L(s''_3) = 1 - \bar{p}(s_3)^{1/2} \approx 0.163$*, resulting in* $\mathbb{P}_L[\varphi'] \approx \mathbf{0.083}$*, which is actually worse than the MB LB.*

The best-possible assignment, however, is the scaled dissociation *(the subject of this paper)* $p_L(t'_2) \approx 0.468$*,* $p_L(t''_2) \approx 0$*,*

$p_L(t_2''') \approx 0.811$, $p_L(s_3') = 0$ *and* $p_L(s_3'') = 0.3$, *thus resulting in* $\mathbb{P}_L[\varphi'] \approx \mathbf{0.122}$. *Hence, we obtain* $0.122 \leq \mathbb{P}[\varphi_{Q_2,D}] \leq 0.180$ *as an approximation. Note that the lower and upper bounds are called "oblivious" here, since the probabilities of the substituted tuple identifiers do not depend on any other variables in the formula [19]. One of our main contributions are these scaled dissociations that complement the optimal upper bounds, and together improve upon existing model-based approximation schemes for both upper and lower bounds.* ◊

(2) Constrained Optimization. For "scaling" (i.e., reweighting) the dissociations of tuple identifiers at the lineage level, we make use of two sets of constraints, indicating how tuple probabilities should be scaled. For the resulting upper bounds, we are already guaranteed to improve over the model-based version. These are proven to be the optimal "oblivious" choice in [21], and hence they do not require further optimization. By contrast, the constraint set for the lower bounds results in an entire continuum of possible assignments (see Example 2), some of which correspond to model-based bounds. We aim to find the *optimal choice* (the scaled dissociation) among the entire continuum. Having a non-linear optimization problem at hand, there is no practical method that can find the (global) optimal solution. Instead, we use gradient descent based methods [4] to find candidate (local) optimal solutions. In our experiments, we never observed getting stuck in a local optimum, however.

(3) Decomposition Scheme. Since a single dissociation step (even when scaled to the best-possible choice of probabilities) only provides a single, static interval for the true marginal probability of each query answer, we additionally employ the *iterative decomposition scheme* used in [16, 17, 35] and integrate this with dissociated lineage formulas and their scaled dissociations. This scheme locates independent subformulas, and further combines this step with Shannon expansions (SE) [46] to eliminate variables with multiple occurrences. After decomposition, a single optimization step is again performed, and this repeats until a desired approximation accuracy threshold is met. By using scaled dissociations instead of simpler one-shot model-based bounds, we open up additional possibilities for configuring this current state-of-the-art B&B approach from [16, 17, 35].

Contributions. In summary, we propose a new *anytime approximation framework* for approximating the marginal probabilities of query answers to non-hierarchical sj-free CQs (hard queries), based on a combination of *optimal oblivious upper bounds* and *scaled dissociations* as lower bounds. Under the hood, *constrained optimization* techniques are in place to find the *optimal choice of weights* for the scaled dissociations at the lineage-level. The framework *generalizes* prior state-of-the-art and is shown to, both theoretically and empirically, *outperform* existing model-based methods. More specifically, we achieve a 42%-81% reduction in aggregate in approximation error for a variety of hard queries over TPC-H, Yago3 and a synthetic benchmark.

2 FORMAL BACKGROUND

We next introduce our basic notions related to probabilistic query evaluation, lineages, dissociations, as well as the influence of variables on those lineages. We loosely follow the notation and concepts of [46] but also include more recent concepts from [14, 16, 17, 34], [19–21] and [27, 39].

Probabilistic Databases & Conjunctive Queries. We consider a fixed relational vocabulary $\Sigma = (R_1, \ldots, R_m)$. A *tuple-independent probabilistic database (PDB)* (D, p) is a database D over Σ, plus a probability function $p : D \to [0, 1]$ associating a probability $p(t)$ to each tuple $t \in D$. A possible world is a subset of D generated by independently including each tuple t in the world with probability $p(t)$. We further associate with each tuple t a binary random variable x_t. Possible worlds can be represented by setting $x_t = 0$ if t does not belong to the respective world, and $x_t = 1$ if it does. Furthermore, when ranging over all possible worlds, the *marginal probability* $\mathbb{P}[x_t = 1] = p(t)$ captures the probability that t is contained in a randomly chosen subset of D. We consider *conjunctive queries* (CQs), i.e., first-order formulas $Q(\mathbf{Y}) = \exists X_1 \ldots \exists X_k \, (g_1 \wedge \cdots \wedge g_m)$ where each atom (or subgoal) g_i represents a relation $R_i(\mathbf{X}_i)$. The variables $X_1, \ldots, X_k$ are called existential variables, while variables in $\mathbf{Y}$ are called the head (or "free") variables . These queries correspond to Select-Project-Join queries in SQL and Relational Algebra [1]. We focus on *self-join-free CQs* ("sj-free CQs") in which the atoms g_i represent distinct relational symbols.

The focus of *probabilistic query evaluation* is to compute the marginal probability $\mathbb{P}[Q(D, p)]$ for a *Boolean query* Q, i.e., to compute the probability that the query evaluates to *true* in a randomly chosen world. We will write $\mathbb{P}[Q]$ for short if (D, p) are clear from the context. We focus on Boolean queries, since the probabilistic query evaluation problem for non-Boolean queries reduces to the Boolean query case in a direct way [46]. Examples for the above concepts were already provided in our Introduction.

Lineage Expressions & Read-Once Form. It is well-known that a CQ can be evaluated over a tuple-independent PDB by first computing the lineage of each query answer (which is a positive Boolean expression over the variables x_t that represents all possible derivations of that answer) and by subsequently evaluating the probabilities of the lineage formula [3, 18, 46]. The general problem of conjunctive-query evaluation has first been shown to be #P-hard in [8]. However, if a lineage formula can be represented in *read-once* (or "single-occurrence") *form* (1OF) [23, 25, 32] (a Boolean formula in which each literal appears exactly once), then its

marginal probability can be computed in polynomial time in the number of literals. Indeed, for a 1OF-formula φ, $\mathbb{P}[\varphi]$ can be calculated in linear time in the size of the formula, by applying the following two rules recursively:

$$\mathbb{P}[\varphi] = \begin{cases} \mathbb{P}[\varphi_1] \cdot \mathbb{P}[\varphi_2] & \text{for } \varphi = \varphi_1 \wedge \varphi_2 \\ \mathbb{P}[\varphi_1] \otimes \mathbb{P}[\varphi_2] & \text{for } \varphi = \varphi_1 \vee \varphi_2 \end{cases}$$

where $p_1 \otimes p_2 := 1 - ((1-p_1)\cdot(1-p_2))$. These rules, referred to as the *independent-and* ($\odot$) and *independent-or* ($\otimes$) rule, respectively, can be applied whenever $vars(\varphi_1) \cap vars(\varphi_2) = \emptyset$ where $vars(\varphi)$ denotes the set of variables in a formula φ. This is trivially satisfied for 1OF formulas.

Shannon Expansions. If φ is not read-once, $\mathbb{P}[\varphi]$ can be computed by applying one or more *Shannon expansions* [46]

$$\varphi \equiv \big(x \wedge \varphi[1/x]\big) \vee \big(\neg x \wedge \varphi[0/x]\big)$$

which decompose φ into mutually exclusive subformulas by substituting a variable x in $\varphi[1/x]$ and $\varphi[0/x]$ with the constants *true* and *false*, respectively. For the resulting decomposition, the following equality then holds

$$\mathbb{P}[\varphi] = p(x) \cdot \mathbb{P}[\varphi[1/x]] + \big(1 - p(x)\big) \cdot \mathbb{P}[\varphi[0/x]]$$

which we will refer to as the *exclusive-or* rule ($\oplus$). We note that repeated Shannon expansions will eventually result in a 1OF representation of φ, however at the cost of potentially doubling the size of the formula at each such expansion.

Lineage-Level Dissociations. Let φ and φ' be two Boolean formulas with variables $\mathbf{x}$ and $\mathbf{x}'$, respectively. We say that φ' is a *dissociation* of φ if there exists a substitution $\theta : \mathbf{x}' \to \mathbf{x}$ such that $\varphi'[\theta] = \varphi$. If $\theta^{-1}(x) = \{x'_1, \ldots, x'_d\}$, then we say that variable x *dissociates into* d variables $x'_1, \ldots, x'_d$. Of particular interest in this paper are *1OF dissociations*, i.e., dissociations φ' which are in 1OF. Every lineage has a unique 1OF dissociation up to renaming of variables. Specializing previous results on oblivious bounds [19] to 1OF dissociations, the following property is key to our approach:

Theorem 1. *[19] Let φ be a lineage of a sj-free CQ, and let $\varphi'[\theta]$ be its 1OF dissociation for a substitution θ. Consider a probability function $p : vars(\varphi) \to [0,1]$. Then,*
(1) for functions $p_U : vars(\varphi') \to [0,1]$ such that

$$p_U(x') \geq p(x)$$

with $x \in vars(\varphi)$ and $x' \in \theta^{-1}(x)$, and
(2) for functions $p_L : vars(\varphi') \to [0,1]$ such that

$$\bigotimes_{x' \in \theta^{-1}(x)} p_L(x') \leq p(x) \qquad (1)$$

with $x \in vars(\varphi)$,
it holds that

$$\mathbb{P}_L[\varphi'] \leq \mathbb{P}[\varphi] \leq \mathbb{P}_U[\varphi']$$

where $\mathbb{P}_L[\varphi']$ and $\mathbb{P}_U[\varphi']$ are obtained by assigning p_L and p_U to the variables in φ', respectively.

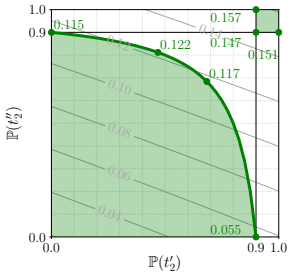


Figure 2: Example 3: The green and gray values show the marginal probability of ψ' for varying $p(t'_2)$ and $p(t''_2)$.

The upper and lower bounds (UBs and LBs) obtained here are called "oblivious", since the probabilities assigned by p_L and p_U to the variables in $\theta^{-1}(x)$ depend only on $p(x)$ and not on the probability of any other variable in φ. Observe that $\mathbb{P}_L[\varphi']$ and $\mathbb{P}_U[\varphi']$ can be computed in PTIME. Furthermore, the *best oblivious UB* p_U is obtained by setting $p_U(x') = p(x)$ for all $x \in vars(\varphi)$ and $x' \in \theta^{-1}(x)$ [19]. By contrast, any function p_L satisfying the inequalities (1) results in a LB for $\mathbb{P}[\varphi]$. The best such lower bounds are those that satisfy (1), in which the inequality is replaced by equality [19].

Model-based Approximations. A *model-based approximation* (MB) of φ consists of two 1OF formulas, φ_U and φ_L, obtained from φ by assigning to one occurrence of each variable the original probability of the variable and assigning all other occurrences of the variable the value *true*, for φ_U, and the value *false*, for φ_L (recall Example 2). It is known that $\mathbb{P}[\varphi_L] \leq \mathbb{P}[\varphi] \leq \mathbb{P}[\varphi_U]$ [16, 17]. It is readily verified that these MB bounds can be seen as special instances of the more general dissociation-based bounds. Indeed, one can first "fully" dissociate φ into φ', such that each variable x in φ dissociates into d_x variables $x'_1, \ldots, x'_{d_x}$ in φ' where d_x is the number of occurrences of x in φ. Furthermore, setting variables to *true* (resp. *false*) in φ_U (resp. φ_L) can be simulated by assigning probabilities 1 (resp. 0) to some dissociated variables in φ'. One obtains functions p_U and p_L in this way such that $\mathbb{P}_U[\varphi'] = \mathbb{P}[\varphi_U]$ and $\mathbb{P}_L[\varphi'] = \mathbb{P}[\varphi_L]$.

Example 3. *Consider a lineage ψ and its 1OF dissociation:*

$$\psi = u_1(r_1(s_1t_1 \vee s_2t_2)) \vee u_2r_3s_3t_2$$
$$\psi' = u_1(r_1(s_1t_1 \vee s_2t'_2)) \vee u_2r_3s_3t''_2$$

As an illustration of Theorem 1, Fig. 2 depicts the space of possible choices of p_L and p_U for the substituted tuples t'_2, t''_2. As for LBs, model-based (MB) bounds are either 0.055 or 0.115, the symmetric-dissociation (SD) bound (which assigns t'_2 and t''_2 equal probability) is 0.117, and the optimal dissociation bound is 0.122. All other valid lower bounds lie in the curved shaded area with the best ones residing on the boundary (bold curved line). As for UBs, MB bounds return either 0.151 or 0.157, while the dissociation-based bound is 0.147. We note

that $0.122 \leq \mathbb{P}[\psi] \approx \mathbf{0.140} \leq 0.147$ *(bounds are rounded) in accordance with Theorem 1.* ◇

Influence. The notion of *influence* quantifies the impact of changes in the probabilities of variables on the marginal probability of a Boolean formula φ. Formally, let $\mathbb{F}_\varphi(\mathbf{x})$ be a function from $\mathbb{R}^n$ to $\mathbb{R}$ where n is the number of variables in φ and $\mathbf{x} \in [0,1]^n$. Let $\mathbb{F}_\varphi(\mathbf{x}) = \mathbb{P}[\varphi]$ where $\mathbf{x}$ is interpreted as a function associating a probability $p(x)$ with each of the n variables in φ. The influence $infl_{x,\varphi}(\mathbf{x})$ of variable x on $\mathbb{P}[\varphi]$ is then defined as follows [27, 39]:

$$infl_{x,\varphi}(\mathbf{x}) := \frac{\partial \mathbb{F}_\varphi(\mathbf{x})}{\partial x}$$

It is known that $infl_{x,\varphi}(\mathbf{x}) = \mathbb{P}[\varphi[1/x]] - \mathbb{P}[\varphi[0/x]]$. As before, $infl_{x,\varphi}(\mathbf{x})$ can be efficiently computed when φ is in 1OF [27]. In general however, computing $infl_{x,\varphi}(\mathbf{x})$ inherits the #P-hardness from computing $\mathbb{P}[\varphi]$.

3 PROBLEM STATEMENT

We are interested in approximating the marginal probability $\mathbb{P}[\varphi]$ of a positive Boolean lineage φ expression to arbitrary precision. We follow the notions of [16]. Given a threshold $\varepsilon > 0$, find a probability $\hat{p}$ that is a *relative ε-approximation*[1] of $\mathbb{P}[\varphi]$, such that the following inequalities hold:

$$(1-\varepsilon) \cdot \mathbb{P}[\varphi] \leq \hat{p} \leq (1+\varepsilon) \cdot \mathbb{P}[\varphi]$$

Given a pair of probabilities (L, U) that fulfill the two conditions (i) $\mathbb{P}[\varphi] \in [L, U]$ and (ii) $(1-\varepsilon) \cdot U \leq (1+\varepsilon) \cdot L$, then any value $\hat{p} \in [(1-\varepsilon) \cdot U, (1+\varepsilon) \cdot L]$ is a relative ε-approximation of $\mathbb{P}[\varphi]$. We call a pair (L, U) an *ε-valid interval bound for* $\mathbb{P}[\varphi]$ and $\frac{U-L}{L+U}$ the interval's *approximation error.*

Our approach is to iteratively obtain ε-valid interval bounds, as inspired by earlier works on anytime approximation algorithms for Boolean lineage expressions [14, 16, 17]. Two key steps underly our method: (1) an *approximation* of $\mathbb{P}[\varphi]$ and (2) an iterative *decomposition* of lineages.

(1) Approximation. We make calls to an approximation "module" that takes as input a lineage φ and returns an interval $[L, U]$ (not necessarily ε-valid) containing $\mathbb{P}[\varphi]$.

In prior work [14, 16, 17], the interval $[L, U]$ is obtained by a model-based approximation φ_L and φ_U of φ. Recall that MB approximations need to make choices of *which* variable occurrences in φ to retain in φ_L and φ_U. Different choices may result in different intervals, and choosing the "best" occurrences of variables to retain is a combinatorial problem: if there are n variables that occur d times each, then there are d^n different choices (e.g., for lineages from queries over the Yago3 dataset, $n = 1\,225$ and $\text{avg}(d) \approx 5.61$, see Fig. 5). Trying all of them to obtain the best interval is not possible.

[1] While we focus here on the most widely used "relative approximation", the approach also extends to the simpler absolute ε-approximation [16] or more advanced approximations like odds ratios [21] in an obvious way.

Also recall that MB bounds can be seen as special instances of more general dissociation-based bounds [19].

These observations motivate us to replace previously used MB bounds with better dissociation-based bounds. We associate with φ its 1OF dissociation φ' and let $L := \mathbb{P}_L[\varphi']$ and $U := \mathbb{P}_U[\varphi']$, where p_L and p_U are as in Theorem 1. Recall that the *upper bound* (UB) $\mathbb{P}_U[\varphi']$ associated with the uniquely defined p_U from dissociation is shown to be *tighter than any MB bound* [19]. By contrast, there is a whole continuum of choices for p_L, each of which results in a different lower bound (LB) (recall Examples 2 and 3). Our problem then is to find the best such p_L among all dissociation-based oblivious LBs for a given lineage. This problem (i) becomes a *continuous optimization problem* instead of a combinatorial one, and (ii) when solved exactly, it will always result in improvements over MB bounds. We refer to these new, *not necessarily model-based*, bounds as *scaled dissociations* and develop optimization techniques in Sect. 4 to guide us towards these scaled dissociations.

Problem 1 (Scaled dissociation). *Given a positive Boolean formula φ, its 1OF dissociation φ', and a probability function p, find a scaled dissociation, i.e., a probability function p_L, defined on the variables of φ', such that $\mathbb{P}_L[\varphi']$ is an oblivious LB of $\mathbb{P}[\varphi]$ and $\mathbb{P}_L[\varphi']$ is maximal among all such LBs.*

(2) Decomposition (or Expansion). There is a limit to how well a given lineage φ can be approximated by model- or dissociation-based bounds. Any resulting interval represents only a single, static "snapshot" of bounds, and we need iterative decomposition techniques to reduce the approximation error further whenever our bounds do not suffice.

In the model-based setting [16, 17], φ is decomposed (i.e., Shannon expanded) into an equivalent form $(x \wedge \varphi[1/x]) \vee (\neg x \wedge \varphi[0/x])$ for some variable x in φ. Then, the probabilities of $\varphi[1/x]$ and $\varphi[0/x]$ are further approximated by MB bounds. This is repeated until an ε-valid interval bound is obtained. In Sect. 5, we show that this approach generalizes to obtain dissociation-based lower and upper bounds, by decomposing the dissociated lineages and carefully adjusting the probability functions p_L and p_U.

Problem 2 (Dissociation-based Anytime Approximation). *Find an anytime approximation algorithm that integrates dissociation-based bounds with a Shannon expansion-based decomposition.*

4 FINDING SCALED DISSOCIATIONS

We first approach Problem 1 by phrasing it as a constrained (i.e., non-linear) optimization problem based on the lower-bound characterization given in Theorem 1 (Sect. 2). A direct encoding of this problem results in an optimization problem

with non-convex objective function and constraints, thus preventing the application of standard gradient-descent based optimization techniques. Nevertheless, we show that after a change of variables, the constraint set can be assumed to be convex (Sect. 4.1). In Sect. 4.2, we then recall two well-known gradient-based optimization techniques, *projected gradient descent* (PGD) and *conditional gradient descent* (CGD) [4]. Our key insight is that both of these methods can be run with either *fixed* or *diminishing step sizes*, thereby alleviating the need to re-evaluate probabilities multiple times in each step (Appendix A). This is important in our setting, since computing marginals, although in PTIME for 1OF lineages, can be time-consuming due to the size of the lineages involved in practice (see Fig. 5).

4.1 Optimization Problem

We start by rephrasing the problem of finding a scaled dissociation as a constrained optimization problem. Let $\varphi'(\mathbf{x}')$ be the 1OF dissociation of $\varphi(\mathbf{x})$ for substitution θ, i.e., $\varphi'[\theta] = \varphi$. Consider the function $F_{\varphi'} : [0,1]^m \to [0,1]$, defined as $F_{\varphi'}(\mathbf{x}') := \mathbb{P}[\varphi'(\mathbf{x}')]$ where m denotes the number of variables in $\mathbf{x}'$. On input values $\mathbf{q} \in [0,1]^m$, $F_{\varphi'}(\mathbf{q})$ returns the probability $\mathbb{P}_{\mathbf{q}}[\varphi'(\mathbf{x}')]$ where $\mathbf{q}$ is regarded as a probability function, i.e., $\mathbf{q}(x'_i) = q_i$. Since φ' is in 1OF, the independent-and and independent-or rules for computing $\mathbb{P}_{\mathbf{q}}[\varphi'(\mathbf{x}')]$ can also be applied when $\mathbf{q}$ takes values outside $[0,1]^m$. We can therefore regard $F_{\varphi'}$ as a function from $\mathbb{R}^m$ to $\mathbb{R}$.

As stipulated by Theorem 1, to find a scaled dissociation, we need to maximize $F_{\varphi'}(\mathbf{x}')$ for valid values of $\mathbf{x}'$. In particular, for each variable x_i in φ, we define a constraint

$$C_i := \big\{\mathbf{q} \in [0,1]^{d_i} \mid \otimes_{j\in[d_i]} q_j = p(x_i)\big\},$$

where d_i is the number of fresh variables that each x_i is dissociated into. In other words, C_i corresponds to probability assignments to the variables in $\theta^{-1}(x_i)$ for which the lower bound condition of Theorem 1 (with equality) holds for variable x_i. Hence, finding a scaled dissociation pours down to finding a probability assignment $\mathbf{p}_L$, such that

$$\mathbf{p}_L := \operatorname{argmax}\big\{F_{\varphi'}(\mathbf{q}_1,\ldots,\mathbf{q}_n) \mid \mathbf{q}_i \in C_i, i \in [n]\big\}. \quad \text{(SLB)}$$

The resulting constraint sets C_i are *non-convex* and therefore prevent the application of standard optimization techniques. A simple change of variables, however, allows us to rephrase the former optimization problem (SLB) as an optimization problem over *convex* sets. Indeed, observe that $\mathbf{q} \in C_i$ iff, for each $j \in [d_i]$, there exists an $\alpha_j \in [0,1]$, s.t. $\bar{q}_j = \bar{p}(x_i)^{\alpha_j}$ and $\sum_{j\in[d_i]} \alpha_j = 1$ (which follows from C_i, as $\bar{q}_1 \cdot \bar{q}_2 \cdots \bar{q}_{d_i} = \bar{p}(x_i)$). We now consider the (convex) probability simplexes

$$\Delta_i := \big\{\boldsymbol{\alpha} \in [0,1]^{d_i} \mid \sum_{j\in[d_i]} \alpha_j = 1\big\},$$

for $i \in [n]$ and the function $G_{\varphi'} : \mathbb{R}^m \to \mathbb{R}$, defined as

$$G_{\varphi'}(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_n) := -F_{\varphi'}\big(1-\bar{p}(x_1)^{\boldsymbol{\alpha}_1},\ldots,1-\bar{p}(x_n)^{\boldsymbol{\alpha}_n}\big),$$

where $1-\bar{p}(x_i)^{\boldsymbol{\alpha}_i}$ denotes the vector $(1-\bar{p}(x_i)^{\alpha_{i1}},\ldots,1-\bar{p}(x_i)^{\alpha_{id_i}})$, for $\boldsymbol{\alpha}_i = (\alpha_{i1},\ldots,\alpha_{id_i})$. Finding a scaled dissociation is now equivalent to solving the following optimization problem over convex sets:

$$\boldsymbol{\alpha}^\star := \operatorname{argmin}\big\{G_{\varphi'}(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_n) \mid \boldsymbol{\alpha}_i \in \Delta_i, i \in [n]\big\}. \quad \text{(SLB')}$$

Indeed, a solution $\mathbf{p}_L$ of (SLB) corresponds to a solution $\boldsymbol{\alpha}^\star$ of (SLB'), with the relationship between the two given by $\mathbf{p}_L = (1-\bar{p}(x_1)^{\boldsymbol{\alpha}^\star_1},\ldots,1-\bar{p}(x_n)^{\boldsymbol{\alpha}^\star_n})$, and vice versa. Thus:

Proposition 1. *A scaled dissociation can be found by solving the latter optimization problem (SLB').*

4.2 Gradient Descent Methods

The optimization of non-convex functions over convex sets is widely studied (see, e.g., [4] for an overview). We focus here on two standard techniques. We let $\Delta = \Delta_1 \times \cdots \times \Delta_n$.

Projected Gradient Descent. Starting from an initial point $\boldsymbol{\alpha}^{(0)}$ in Δ, this method generates a sequence of points $\boldsymbol{\alpha}^{(t)}$ in Δ. Given $\boldsymbol{\alpha}^{(t)}$, the next point $\boldsymbol{\alpha}^{(t+1)}$ is obtained by first moving along the gradient direction, i.e., $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\alpha}^{(t)} - \eta_t \nabla G_{\varphi'}(\boldsymbol{\alpha}^{(t)})$ for some step size η_t, followed by a *projection* of $\boldsymbol{\beta}^{(t+1)}$ on Δ, resulting in $\boldsymbol{\alpha}^{(t+1)} \in \Delta$. In particular,

$$\boldsymbol{\alpha}^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\Delta} \|\boldsymbol{\alpha} - \boldsymbol{\beta}^{(t+1)}\|_2,$$

i.e., $\boldsymbol{\alpha}^{(t+1)}$ is the point in Δ closest to $\boldsymbol{\beta}^{(t+1)}$ relative to the Euclidean norm $\|\cdot\|_2$. While such a projection is in generally complex to compute, the projection on *simplexes* Δ_i and Δ can however be computed efficiently [48].

Conditional Gradient Descent (Frank-Wolfe). As in PGD, a sequence of points $\boldsymbol{\alpha}^{(t)}$ is generated, starting from $\boldsymbol{\alpha}^{(0)}$. By contrast to PGD, projection is avoided by considering a linear combination of $\boldsymbol{\alpha}^{(t)} \in \Delta$ and $\boldsymbol{\gamma}^{(t)} \in \Delta$, i.e., $\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \eta_t(\boldsymbol{\gamma}^{(t)} - \boldsymbol{\alpha}^{(t)})$, for some step size η_t. Furthermore, $\boldsymbol{\gamma}^{(t)} \in \Delta$ is found by solving another optimization problem:

$$\boldsymbol{\gamma}^{(t)} = \operatorname*{argmin}_{\boldsymbol{\gamma}\in\Delta} \nabla G_{\varphi'}(\boldsymbol{\alpha}^{(t)})' \cdot (\boldsymbol{\gamma} - \boldsymbol{\alpha}^{(t)}).$$

We again benefit from having simplexes as constraint sets. Indeed, on simplexes Δ, such a $\boldsymbol{\gamma}^{(t)}$ can be efficiently computed (see Example 3.2.1 in [4]). Indeed, recall that $\boldsymbol{\gamma}^{(t)} = (\boldsymbol{\gamma}_1^{(t)},\ldots,\boldsymbol{\gamma}_n^{(t)})$ where $\boldsymbol{\gamma}_i^{(t)}$ is a d_i-dimensional vector corresponding to variables in $\theta^{-1}(x_i)$. It is known that setting $\gamma_{ik}^{(t)} = 1$, for each $i \in [n]$, when k is an index that minimizes $\frac{\partial G_{\varphi'}}{\partial \alpha_{ik}}(\boldsymbol{\alpha}^{(t)})$, and $\gamma_{i\ell}^{(t)} = 0$, for $\ell \in [d_i]$ distinct from k, results in a solution $\boldsymbol{\gamma}^{(t)}$ of the above optimization problem. In other words, the optimum $\boldsymbol{\gamma}^{(t)}$ needs to be a corner point in the simplex.

In both PGD and CGD, we can take as initial point $\boldsymbol{\alpha}^{(0)}$, e.g., the "symmetric" point in which, for each $i \in [n]$ and $j \in [d_i]$, we set $\alpha_{ij}^{(0)} := 1/d_i$. This corresponds to the symmetric dissociation-based lower bound, as described in Example 2. We also point out that these methods provide points in the simplex Δ in each step. This implies that in every step a corresponding lower bound $\mathbb{P}_L[\varphi']$ for $\mathbb{P}[\varphi]$ is obtained.

A well-known issue with gradient-based methods (or any other methods for that matter) for solving non-linear optimization problems is that they may converge to a local optimum rather than to the desired global optimum. As such, our methods may not necessarily find the desired optimal scaled dissociation. This implies that we may not necessarily improve on MB lower bounds. Yet, we did not observe this behaviour in our experiments. We also note that finding the scaled dissociations is only one part of our approximation scheme. Even if we were to be stuck in a local optimum, subsequent Shannon expansions may get us back on track.

5 ANYTIME APPROXIMATION ALGORITHM (AAA)

So far, we know how to obtain a pair of probability functions p_L and p_U, s.t. $\mathbb{P}_L[\varphi'] \leq \mathbb{P}[\varphi] \leq \mathbb{P}_U[\varphi']$, for a given 1OF dissociation φ' of a positive Boolean formula φ'. This yields a single, static interval of lower and upper bounds which cannot be further improved by using dissociations alone.

We next describe how dissociation-based approximations can be gracefully embedded into an *incremental* lineage-compilation algorithm which can approximate $\mathbb{P}[\varphi]$ to arbitrary precision. The compilation techniques are inspired by previous anytime approximation frameworks [14, 16, 17], yet generalize and substantially improve them. The core of our resulting *anytime approximation algorithm* (AAA) is depicted in Algorithm 1. It contains the three procedures, (1) `OptimizeBounds`, (2) `PickLeaf` and (3) `SelectVariable`, which can be instantiated in various ways.

d-Trees. AAA uses the same representation of lineages as [16], so-called "decomposition trees" (*d-trees*, for short). A d-tree is simply a formula constructed with $\otimes$ (*independent-or*), $\odot$ (*independent-and*), and $\oplus$ (*exclusive-or*) as the inner nodes of the tree, while propositional formulas form the leaves. Any propositional formula can be recursively compiled into a d-tree, as shown in Lines 13 and 14 for $\otimes$, Lines 15 and 16 for $\odot$, and Lines 17 to 20 for $\oplus$. More specifically, the $\oplus$-decomposition involves the selection of a leaf ψ and a variable $x \in vars(\psi)$ (procedures `PickLeaf` and `SelectVariable` in Lines 17 and 18) followed by the Shannon expansion $(x \odot \psi[1/x]) \oplus (\neg x \odot \psi[0/x])$ (Line 20). In the following, φ interchangeably denotes a lineage or a d-tree; this will always be clear from the context. Lower and upper bounds on the probabilities of the leaves (i.e., formulas) can be propagated

Algorithm: AAA (Anytime Approximation Algorithm)
Input: d-tree φ in which each leaf ψ is associated with a 1OF dissociation ψ' and probability functions p_L and p_U, accuracy bound ε, probability function p
Output: ε-valid interval bound $[L, U]$

1 $P_L \leftarrow \emptyset$; $P_U \leftarrow \emptyset$
2 **if** *Global* **then**
3 **for** each leaf ψ **do**
4 $p'_L, p'_U \leftarrow$ `OptimizeBounds`$(\psi, \psi', p_L, p_U, p)$
5 $P_L \leftarrow P_L \cup \{p'_L\}$; $P_U \leftarrow P_U \cup \{p'_U\}$;
6 **else if** *Local* **then**
7 Let ψ:= `PickLeaf`(φ, p)
8 $p'_L, p'_U \leftarrow$ `OptimizeBounds`$(\psi, \psi', p_L, p_U, p)$
9 $P_L \leftarrow P_L \cup \{p'_L\}$; $P_U \leftarrow P_U \cup \{p'_U\}$;
10 $L \leftarrow$ `PropagateUp`(φ, P_L); $U \leftarrow$ `PropagateUp`(φ, P_U)
11 **if** $(1-\varepsilon) \cdot U < (1+\varepsilon) \cdot L$ **then return** $[L, U]$
12 **while** $\exists$ *leaf* ψ *that is decomposable* **do**
13 **if** $\exists\, \psi_1, \psi_2$, *s.t.* $(\psi = \psi_1 \vee \psi_2)$ & $(vars(\psi_1) \cap vars(\psi_2) = \emptyset)$ **then**
14 For $Y \in \{\psi, \psi', p'_L, p'_U\}$: replace Y with $Y_1 \otimes Y_2$
15 **if** $\exists\, \psi_1, \psi_2$, *s.t.* $(\psi = \psi_1 \wedge \psi_2)$ & $(vars(\psi_1) \cap vars(\psi_2) = \emptyset)$ **then**
16 For $Y \in \{\psi, \psi', p'_L, p'_U\}$: replace Y with $Y_1 \odot Y_2$
17 Let ψ:=`PickLeaf`(φ, p)
18 Let x:=`SelectVariable`(ψ)
19 Let $\psi_1 = \psi[1/x]$ and $\psi_2 = \psi[0/x]$
20 For $Y \in \{\psi, \psi', p'_L, p'_U\}$: replace Y with $(x \odot Y_1) \oplus (\neg x \odot Y_2)$
21 **return** AAA$(\varphi, \varepsilon, p)$ (Note: φ is the updated d-tree)

Algorithm 1: Recursive anytime approx. algorithm.

to bounds on the entire d-tree (representing the original lineage φ) by means of the three computation rules given earlier, i.e., $\mathbb{P}_q[\varphi_1 \odot \varphi_2] = \mathbb{P}_q[\varphi_1] \cdot \mathbb{P}_q[\varphi_2]$, $\mathbb{P}_q[\varphi_1 \otimes \varphi_2] = 1-(1-\mathbb{P}_q[\varphi_2]) \cdot (1-\mathbb{P}_q[\varphi_2])$, and $\mathbb{P}_q[\varphi_1 \oplus \varphi_2] = \mathbb{P}_q[\varphi_1] + \mathbb{P}_q[\varphi_2]$, where q refers to either p_L or p_U. To approximate $\mathbb{P}[\varphi]$, it thus suffices to compile φ into a d-tree, s.t. the probability of each of the leaves can be approximated well enough. One may safely stop when all leaves are 1OF, as then the exact probability can be computed in linear time in the size of the d-tree, which however may be exponentially large. In Algorithm 1, procedure `PropagateUp` takes care of the computation of the bounds L and U (Line 10), based on the structure of d-tree φ and the collections P_L and P_U of probability functions associated with the 1OF dissociations at the leaves.

Incremental Compilation. Bringing the dissociations into picture is fairly simple now. Not only do we compile φ into a d-tree, but also during φ's decomposition we carry along a 1OF dissociation φ' of φ (for substitution θ) and probability functions p_L and p_U defined over the variables in φ'. We decompose these together with φ. More specifically, when a leaf ψ decomposes in $\psi_1 \odot \psi_2$ or $\psi_1 \otimes \psi_2$, then it is easily verified that we can also decompose $\psi' = \psi'_1 \odot \psi'_2$ or $\psi' = \psi'_1 \otimes \psi'_2$, respectively, where ψ'_i is a 1OF dissociation of

It	Step	d-Tree	Associated 1OF dissociation	$\approx L$	$\approx U$	$\frac{U-L}{L+U}$	bound
1	Initial	$\varphi_{Q_2,D} = u_1(r_1(s_1t_1 \vee s_2t_2) \vee r_2s_3t_2) \vee u_2r_3s_3t_2$	$\varphi'_{Q_2,\mathcal{D}} = u_1(r_1(s_1t_1 \vee s_2t'_2) \vee r_2s'_3t''_2) \vee u_2r_3s''_3t'''_2$	0.114	0.274	0.41	MB
	Optimize	$\varphi_{Q_2,D} = u_1(r_1(s_1t_1 \vee s_2t_2) \vee r_2s_3t_2) \vee u_2r_3s_3t_2$	$\varphi'_{Q_2,\mathcal{D}} = u_1(r_1(s_1t_1 \vee s_2t'_2) \vee r_2s'_3t''_2) \vee u_2r_3s''_3t'''_2$	0.083	0.180	0.37	PGD
				0.122	0.180	0.19	PGD
2	Expand	$\varphi_{Q_2,D} = (s_3 \odot \varphi[1/s_3]) \oplus (\neg s_3 \odot \varphi[0/s_3])$	$\varphi'_{Q_2,\mathcal{D}} = (s_3 \odot \varphi'_1) \oplus (\neg s_3 \odot \varphi'_2)$	0.139	0.176	**0.11**	MB
	Optimize	$\varphi_{Q_2,D} = (s_3 \odot \varphi[1/s_3]) \oplus (\neg s_3 \odot \varphi[0/s_3])$	$\varphi'_{Q_2,\mathcal{D}} = (s_3 \odot \varphi'_1) \oplus (\neg s_3 \odot \varphi'_2)$	0.122	0.172	0.17	PGD
				0.146	0.172	**0.08**	PGD

Figure 3: Running Example 4: Run of AAA with PGD (blue) or MB (red) on $\varphi_{Q_2,D}$ and $\varphi'_{Q_2,\mathcal{D}}$ from Examples 1 and 2.

ψ_i, for $i = 1, 2$. Similarly, for p_L and p_U; these get split into p_L^1 and p_U^1 for ψ'_1, and p_L^2 and p_U^2 for ψ'_2, by simply restricting p_L and p_U to the appropriate sets of variables. When ψ is Shannon-expanded into $(x \odot \psi[1/x]) \oplus (\neg x \odot \psi[0/x])$, we consider $\psi'_1 = \psi'[1/\theta^{-1}(x)]$ and $\psi'_2 = \psi'[0/\theta^{-1}(x)]$, obtained from ψ' by setting all variables in $\theta^{-1}(x)$ to true or false, respectively, and thereby obtain 1OF dissociations of $\psi[1/x]$ and $\psi[0/x]$. We furthermore ensure that ψ'_1 and ψ'_2 have *no variables in common*, by applying an additional substitution. This is to ensure that each leaf can be optimized separately. Again, p_L and p_U are decomposed by restricting variables (and possibly rescaling) whenever one or more occurrences of variables other than x are removed by the expansion. It can be verified (see Appendix C) that, when starting from a 1OF lineage φ' of φ and functions p_L and p_U satisfying the conditions of Theorem 1, then these conditions remain satisfied after each decomposition step, thus justifying the following proposition.

Proposition 2. *In every step of the algorithm, every leaf ψ in the d-tree has a 1OF-dissociation ψ' and probability functions p_L and p_U associated with it, such that $\mathbb{P}_L[\psi'] \leq \mathbb{P}[\psi] \leq \mathbb{P}_U[\psi']$ holds.*

We thus obtain an approximation of $\mathbb{P}[\varphi]$ after each decomposition step by recursively calling `PropagateUp`. Furthermore, we can show that L and U can never deteriorate by applying any of the three decomposition steps (again see Appendix for details).

Obtaining ε-Valid Bounds. Algorithm AAA continues decomposing lineages, associated dissociations, and probability functions until an ε-valid interval bound for $\mathbb{P}[\varphi]$ is obtained (Line 11). Furthermore, AAA allows an additional optimization of the probability functions for the dissociations in the leaves (procedure `OptimizeBounds`, Lines 4 and 8). Intuitively, this is to accommodate for finding the scaled dissociation (lower bound) after each decomposition step, but other optimization methods (such as model-based) can be plugged in as well (see the next section). Furthermore, we allow either a local or **`global`** (`G`) optimization, where a global optimization calls `OptimizeBounds` on every leaf (Lines 2 to 5), whereas a **`local`** (`L`) optimization uses `PickLeaf` to pick a single leaf to optimize (Lines 6 to 9). To guarantee that we gradually move towards an ε-valid bound and keep the bounds valid at all times, we require the updated functions from `OptimizeBounds` to satisfy the conditions in Theorem 1 and only accept them if they lead to an improvement in the quality of the approximation. Since—in the worst case—we eventually end up with a d-tree, in which all leaves are 1OF, we are guaranteed to generate an ε-valid interval bound for $\mathbb{P}[\varphi]$. Example 4 illustrates the algorithm and demonstrates that choosing scaled dissociation lower bounds may lead to a valid bound faster than using model-based bounds.

Example 4. *In Fig. 3, we list the different steps of* AAA*, starting with $\varphi_{Q_2,D}$ and $\varphi'_{Q_2,\mathcal{D}}$ from Examples 1 and 2, and for $\varepsilon = 0.1$. Blue-shaded entries for L, U and approximation error $\frac{U-L}{L+U}$ correspond to applying projected gradient descent (PGD) during the optimization step; p_U is kept fixed as the optimal oblivious upper bound (UB), whereas p_L is initialized by the symmetric dissociation-based lower bound (SD LB) (see Example 2). Then Algorithm* AAA *calls* `OptimizeBounds` *using PGD to find the best scaled dissociation LB for the given dissociated lineage expression. The error decreases from* 0.37 *to* 0.19*, but is not yet below ε. Next* AAA *performs a Shannon expansion (iteration 2), here on s_3, resulting in a new dissociated lineage with improved UB and error of* 0.17*. We pass the leaves of the expanded d-tree to* `OptimizeBounds`*. Since $\varphi[0/s_3]$ is in 1OF and needs no further approximation, only $\varphi[1/s_3]$ is optimized for its LB for its dissociation φ'_1 using again PGD. Propagating the obtained bounds for $\varphi[1/s_3]$ back up, we get $L \approx 0.146$, $U \approx 0.172$, and an error of $0.08 < \varepsilon$. Hence, (L, U) is an ε-valid interval bound for the true yet unknown $\mathbb{P}[\varphi_{Q_2,D}] \approx 0.170$.*

As comparison, we also show in red the trace of AAA *when the best model-based (MB) UBs and LBs are used, which are one-shot and no optimization is possible. After one Shannon expansion (iteration 2), we obtain an error of* 0.11*, still above ε. As a consequence, a additional Shannon expansion is needed (on t_2, not shown). This results in a d-tree in which all leaves are 1OF. As a consequence, the exact probability $\mathbb{P}[\varphi_{Q_2,D}]$ will be returned, at the expense of an extra Shannon expansion.* ◇

Runtime Optimizations. We apply the following runtime optimizations to further accelerate the execution of Algorithm 1, by decreasing the size of the leaf formulas.

- If there exists a sub-formula χ at a leaf ψ which only contains variables with single occurrences, we replace the entire sub-formula with a new variable x so that $p(x) = \mathbb{P}(\chi)$. We can calculate $\mathbb{P}(\chi)$ exactly, since the sub-formula is in 1OF.

Procedure	Decisions	Choices
OptimizeBounds	Method	MB, SD, PGD_d, CGD_d
	#Steps	1, 10
	Strategy	local (L), global (G)
SelectVariable	Selection	Occmax(O), Imax(I), Weighted Imax(WI)

Figure 4: Overview of all instantiations of our framework. We use *Method/Selection/#Steps/Strategy* to denote specific configurations for the decisions. For example, CGD_d/I/10/G uses CGD_d as optimization method, Imax to select variables for Shannon expansion, performs up to 10 steps of CGD_d before the next decomposition, and uses the global optimization strategy. We also allow MB (which originally did not have multiple optimization steps) to run repeatedly and return the best model-based bound found. For example, MB/O/10/L picks the best from 10 randomly chosen MB bounds. By contrast, the SD method is inherently single-step and global. This results in a total of 39 different instantiations. ($3 \times 2 \times 2 \times 3 = 36$ for MB, PGD, CGD; $1 \times 1 \times 1 \times 3 = 3$ for SD.)

This does not affect $\mathbb{P}(\psi)$, but may substantially decrease the size of the overall formula.

• The second reduction deals with implications. If a leaf contains conjunctions (resp. disjunctions) which are always *false* (resp. *true*), the entire conjunction (resp. disjunction) is replaced with *false* (resp. *true*). This can occur due to the substitution in Line 20 of AAA. We thereby again reduce the sizes of the leaves. A possible side-effect is that occurrences of other variables are eliminated as well. To ensure that we still have valid lower bounds, the probability mass of the removed occurrences is evenly re-distributed among the remaining occurrences.

Both of these run-time optimizations are performed exhaustively at the start of the algorithm (for the first optimization) and after each SE step, to reduce the size of the lineage φ as much as possible.

6 INSTANTIATIONS OF AAA

The algorithm AAA provides a framework in which different strategies for procedures (1) OptimizeBounds, (2) PickLeaf and (3) SelectVariable can be plugged in. We next describe several such instantiations which will serve as the basis of our experiments.

(1) OptimizeBounds. We next describe four different methods for obtaining lower (LBs) and upper bounds (UBs).

(a) MB uses randomly chosen *model-based bounds* as proposed by [16, 17, 34]. For LBs p'_L: for each variable $x_i \in \mathit{vars}(\psi)$, with d_i dissociated occurrences $x'_{ij} \in \mathit{vars}(\psi')$, pick $k \in [d_i]$ at random, and set $p'_L(x'_{ik}) = p(x_i)$ and $p'_L(x'_{ij}) = 0$ for $j \neq k$. Analogously, for UBs p'_U: pick $k \in [d_i]$ at random, and set $p'_U(x'_{ik}) = p(x_i)$ and $p'_U(x'_{ij}) = 1$ for $j \neq k$.

(b) SD uses the *symmetric dissociations* from [20, 21]. For LBs p'_L: for each variable $x_i \in \mathit{vars}(\psi)$, with d_i dissociated occurrences x'_{ij}, we set $p'_L(x'_{ij}) = 1 - \bar{p}(x_i)^{1/d_i}$. For UBs p'_U: we use the optimal dissociation-based UB, i.e., $p'_U(x'_{ij}) = p(x_i)$.

(c) PGD and **CGD** use optimal dissociation-based UBs, just as in SD. For LBs, however, they use *scaled dissociations*, i.e., an "optimal" p'_L obtained by using the *projected gradient descent* (resp. *conditional gradient descent*) methods. Both methods are initialized by symmetric dissociations and use step sizes determined by the Lipschitz constant (see Appendix A).

Dampening (d). In our experiments, we observed that the Lipschitz constant L is sometimes large, thus leading to very small step sizes. Furthermore, optimal values of $G_{\varphi'}$ are often situated on the boundaries of the simplex Δ. Small step sizes, however, require many iterations to reach these boundaries and thus reduce the performance of the inference algorithm. Motivated by this, we modify the step sizes η_t in PGD and CGD methods by "*damping*," which is inspired by a similar technique used in belief propagation [30]:

• **PGD_d**: To increase the likelihood that points on the boundary of Δ are reached, we move in the first step to a point *outside* Δ (that way, the resulting point lies on the boundary of Δ after the projection). To this aim, instead of using a single step size across all simplexes Δ_i, we consider a vector $\boldsymbol{\eta}_t = (\eta_1^{(t)}, \ldots, \eta_n^{(t)})$ of step sizes, one for each Δ_i. We set the initial step size $\eta_i^{(0)}$ to $1/\max_{k \in [d_i]} \left\{ \frac{\partial G_{\varphi'}}{\partial \alpha_{ik}} \right\}$. This ensures that the next $\boldsymbol{\beta}^{(1)}$ will be pushed outside the simplex and that projection is again necessary. To recover correctness, we use *damping*: We set $\boldsymbol{\eta}^{(t+1)} = \boldsymbol{\eta}^{(t)}$ in case $G_{\varphi'}(\boldsymbol{\alpha}^{(t-1)}) > G_{\varphi'}(\boldsymbol{\alpha}^{(t)})$, i.e., when we move in the "right" direction; otherwise, we dampen the step size: $\boldsymbol{\eta}^{(t+1)} = 3/4 \cdot \boldsymbol{\eta}^{(t)}$. When damping is applied sufficiently many times, we reach a step size smaller than $2/L$, from which correctness follows.

• **CGD_d**: We always start with the maximal step size, i.e., $\eta_0 = 1$, and use damping as just described. We cannot guarantee that step sizes will become small enough after damping, since correct step sizes depend on the gradient at the current point. Nevertheless, experiments show that CGD_d has a positive impact on the overall query approximation.

Both PGD_d and CGD_d start from the symmetric dissociation lower bound as a starting point and use optimal dissociation-based both for the upper bound. Our experiments (not included) show that these variants always outperform their standard counterparts. We therefore only report on PGD_d and CGD_d in our experiments.

(2) PickLeaf. For PickLeaf we use the heuristic also employed in [16], which picks the leaf with the highest difference between upper and lower bound

$$\psi^\star := \operatorname*{argmax}_{\psi} \left(\mathbb{P}_U[\psi'] - \mathbb{P}_L[\psi'] \right),$$

where ψ ranges over all leaves in φ (when seen as a d-tree). We break ties arbitrarily.

(3) `SelectVariable`. We next focus on different choices for selecting the variable used in a Shannon expansion. • **`Occmax`** is used by [16, 17] and selects the variable x in the chosen leaf ψ with the highest number of occurrences. Again, ties are broken arbitrarily. Thus, `Occmax` returns:

$$x := \operatorname*{argmax}_{x \in \psi} d_x$$

where x ranges over all variables in ψ in the current d-tree. Notice that `Occmax` does not use information on probabilities.

• **`Imax`** is a new heuristic selection method based on influence (see Sect. 2). Intuitively, it selects the variable x in a leaf ψ that has the largest sum of influences on the overall probability, summing over its occurrences in ψ's dissociation. We break ties arbitrarily. Thus, `Imax` returns:

$$x := \operatorname*{argmax}_{x \in \psi} \Big(\sum_{x'_i \in \theta^{-1}(x)} infl_{x'_i,\psi'}(\mathbf{p}_L) \Big)$$

• **`Weighted Imax`** is a variant of `Imax` where we multiply the sum of influences of occurrences with the original variable probability. Thus, `Weighted Imax` returns:

$$x := \operatorname*{argmax}_{x \in \psi} \Big(p(x) \cdot \sum_{x'_i \in \theta^{-1}(x)} infl_{x'_i,\psi'}(\mathbf{p}_L) \Big)$$

To conclude, a summary of all methods and how we refer to specific combinations of variants can be found in Fig. 4.

Our experiments will show that all methods work well. Nevertheless, we propose the method $\mathtt{PGD}_d\mathtt{/O/1/G}$ as the **method of choice** since it performs consistently well for all lineages considered.

7 EXPERIMENTS

Our experiments answer one key question: what is the *average relative error over time* for various instantiations of our anytime approximation framework and prior work?

We provide a detailed comparison of our methods that approximate the *best oblivious bounds* ($\mathtt{PGD}_d$ and $\mathtt{CGD}_d$) against two state-of-the-art baselines for approximate inference: the *model-based* method (`MB`) of [16, 17] and the *symmetric dissociation* (`SD`) bounds of [20, 21], which had never been applied at the lineage level. We also explore the impact of different instantiations of our approach (Fig. 4).

We first examine the effect of our different configurations in detail on synthetically generated data, and then validate our results by testing our winning methods against the existing ones on two real-life datasets.

Reproducibility. The code to reproduce our results will be made available at: http://github.com/northeastern-datalab/scaled-dissociations

7.1 Common Experimental Setup

We start by describing the general setup for the experiments.

Datasets & Lineages. We implemented a synthetic generator for queries and datasets (see Sect. 7.2) and use two real-life data sets and corresponding sets of queries: Yago3 [31] and TPC-H [47] using a scale factor of 1 (see Sect. 7.4).

In all these settings, we obtain lineages by enumerating all dissociation query plans and choosing up to *5 plans* at random. Furthermore, we initialize tuple probabilities in the datasets with *15 different random seeds* (5 according to a *uniform* distribution, 5 according to a *normal* distribution with $\mu = 0.25$ and $\sigma = 0.2$, and 5 according to an *exponential* distribution with $\lambda = 1$). All probabilities take values in the range of $(0, 0.5]$. The values of the parameters are chosen so that they provide interesting values for query probabilities. We only consider non-hierarchical sj-free conjunctive queries for which inference is #P-hard, and we excluded any lineages that were already in 1OF at the start, and any lineages whose upper and lower bounds were 1 (due to numerical precision).

All combined, **a total of 4800 such lineages** were constructed from the queries over the three datasets (3060 for the synthetic datasets, 765 for Yago3 and 975 for TPC-H). Further statistics of these lineages are shown in Fig. 5. Note that we count the number of variables as the number of variables with repeated occurrences, as those determine the hardness of a problem (variables that occur only once are reduced to a single constant by the reduce operation and act as constants in the optimization process). Likewise, the number of occurrences is counted only over repeated variables. Number of nodes, however, include all tuple variables.

The query plans run in the order of seconds. While naive probabilistic query plans are slower than standard relational queries (sometimes substantially because they prescribe a particular join order which may not be optimal), it is known that their running time can be brought down close to that of standard deterministic query evaluation by first applying a simple *deterministic semi-join reduction*, and then running the probabilistic plan on the reduced data [21]. Since the focus of this paper is on the quality/runtime trade-off between various approximation methods, we ignore the time consumed for relational processing and the construction of lineage structures, which are common to all approaches.

Experimental Runs. For each of the *39 instantiations* described in Fig. 4 and each lineage, we let `AAA` run for *30 seconds*, storing the obtained relative approximation error ε after each iteration. Here, an iteration consist of a single optimization and expansion (recall Example 4). As described in Sect. 3, the approximation error is given by $\frac{U-L}{U+L}$ which provides the smallest ε for which $[L, U]$ is a guaranteed ε-valid interval bound. We then divide the number of iterations into 30 time windows of 1 second, where the minimal error

	Nodes	Variables	Occurrences	avg(degree)
Synthetic	25,125	1,021	16,442	11.0
Yago3	18,918	1,225	16,735	5.61
TPC-H	41,919	4,035	17,063	5.45

Figure 5: Average numbers of nodes in a tree-representation of the lineage, average number of *repeated* variables and their occurrences, and average degree (occurrences per variable) in the lineage expressions for our 3 datasets.

Parameter	Values
Query type	chain, cycle, star
Query length (k)	3, 4, 5
Domain size (d)	100, 200
Number of tuples	500
Domain distribution	uniform, normal, exponential

Figure 6: Parameters and values for synthetic dataset. In total, we have 54 different configurations ($3 \times 3 \times 2 \times 1 \times 3$).

for each window is kept. The figures below show the average of these minimal errors per window over all lineages for each instantiation. Only one in three points is shown.

Comparison with State-of-the-Art. We do not compare to sampling as SPROUT [16, 17], which is a special case of our method, was already shown to outperform sampling-based methods. However, we do compare to the approach used by SPROUT. More specifically, it is readily verified that SPROUT's approximation method corresponds to instantiation MB/O/1/L in our framework, i.e., it uses a single MB bound, performs local optimization and uses Occmax for variable selection [16, 17]. Our experiments have shown that MB/O/10/L performs better than MB/O/1/L (thus choosing the best among 10 random MB bounds in each iteration is better than choosing just one random MB bound). Thus, MB/O/10/L in the experiments corresponds to an improved state-of-the-art competitor.

General Setup. All experiments are run on an Intel Xeon 32x2.5 GHz server with 32 GB RAM and 16 physical cores, using PostgreSQL 8.3.3 as a storage backend for the data. We implemented all methods in Python 2.7.

7.2 30-Second Runs on Synthetic Data

Queries. Our synthetic generator creates queries and datasets from which lineage formulas are then constructed. We use three *types of queries*, each of which takes a size parameter k to parameterize it. Chain queries are defined as:

$$Q_k^{\infty} :- R_1(X_1), R_2(X_1, X_2), \ldots, R_{k-1}(X_{k-2}, X_{k-1}), R_k(X_{k-1})$$

Q_2 in Example 1 is an instantiation of a chain query with $k = 3$. Cycle queries are defined as:

$$Q_k^{\circ} :- R_1(X_k, X_1), R_2(X_1, X_2), \ldots, R_k(X_{k-1}, X_k)$$

And Star queries are defined as:

$$Q_k^{\star} :- R_0(X_1, \ldots, X_k), R_1(X_1), \ldots, R_k(X_k)$$

Data. For each query, we construct a domain with a fixed *domain size d* for all query variables $X_1, X_2, \ldots$. We then give normalized weights to each element in the domain by sampling from a uniform, normal or exponential *domain distribution*. We then construct tuples for each relation by sampling from the domains of the variables using the respective weights. We found that setting the *number of tuples* in each relation to 500 provided interesting lineages that were neither too small (and therefore trivial) nor too large to manage. Note that only a subset of these tuples is included in the lineage, those that participate in a join. The statistics in Fig. 5 show that we got a good spread of lineage sizes that are comparable to those in the real-life datasets.

Configurations. In total we have *54 different configurations* (see Fig. 6), each of which gets repeated between 25 and 125 times depending on the number of query plans that exist. This leads to *3060 lineages*. We use this synthetic experiment to find out which instantiations of the algorithm (see Fig. 4) work best and what the effect of each of the methods is on the overall efficiency and accuracy.

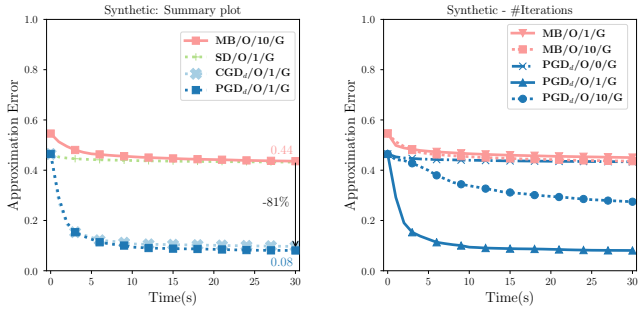
(1) Overall Results. We first focus is on comparing our new optimization-based methods against the existing model-based (MB) and symmetric dissociation (SD) methods.

Our optimization-based methods reduce the approximation error, on average, by 81% on synthetic data.

• Figure 7c shows a confusion matrix that compares the approximation error of different instantiations of AAA and shows in how many cases each method in the row header is at least 10^{-2} (resp. 10^{-4}) better than the one in the column header. For each of the optimization methods (MB, SD, CGD_d,PGD_d), we only report the best configuration for the other decisions. Best here means the method that has the highest sum over the elements in its row in Fig. 7c (i.e., "won" more times than the others in total). As expected, our new optimization methods PGD_d and CGD_d outperform the others with a significant margin. Out of all 3060 lineages, the optimization-based methods win at least 74% (resp. 82%) of the time with a difference of 10^{-2} (resp. 10^{-4}) in approximation error against either of the existing methods, whereas they are almost never beaten themselves.

• Figure 7a shows the approximation error over time for each instantiation: it is clear that the two optimization methods very quickly decrease the error in the beginning, whereas the others taper off almost immediately. On average, the optimization methods decrease the error by 81%.

(2) Impact of instantiation decisions. We compare the impact of the design decisions (Fig. 4) on performance for



(a) (b)

Method	**PGD$_d$/O/1/G**	CGD$_d$/O/1/G	MB/O/10/G	SD/O/1/G
PGD$_d$/O/1/G	–	**26% (44%)**	**75% (83%)**	**75% (82%)**
CGD$_d$/O/1/G	**3% (24%)**	–	75% (83%)	75% (82%)
MB/O/10/G	**0% (0%)**	0% (0%)	–	37% (51%)
SD/O/1/G	**0% (0%)**	0% (0%)	15% (29%)	–

(c)

Figure 7: Results on the synthetic data show a consistent improvement of our methods over prior ones: (a) The approximation error after 30 sec gets reduced by 81%, on average. (b) The number of optimization steps affects our methods considerably. (c) The prior methods never outperform ours: the fraction of lineages for which each method in a row has a smaller approximation error after 30 seconds than the one in the column, with difference at least 10^{-2} (resp. 10^{-4}). For each bounding method, we chose the best instantiations.

PGD$_d$ and MB. We examine one parameter at a time and leave all others to the winning configurations from Fig. 7a.

The optimization-based methods outperform the existing ones significantly across any combination of design decisions. The most important decision is the number of steps, whereas different variable selection methods and local vs. global optimization make little difference.

• The **number of optimization steps** noticeably impacts the performance of the PGD$_d$ method. In Fig. 7b the version that uses 10 iteration steps seems like it will eventually converge to a similar accuracy as the version that uses a single optimization step, but does so at a much slower pace. (To compare with "zero" steps, we also added SD as PGD$_d$/O/0/G, as it is the starting point for PGD$_d$.) This is expected, as the gain in accuracy decreases for each subsequent optimization step, so after a limited number of steps a Shannon expansion brings more improvement than further iterations. For MB, the additional iterations add little improvement. More possibilities are examined, but the choice is still *randomly made* from an exponential number of possible bounds.

• By contrast, the choice of **variable selection** heuristic has limited impact on the overall accuracy of the methods (Fig. 8a): PGD$_d$ performs equally well as Occmax(O), Imax(I) or Weighted Imax(WI). Whereas we expected the Imax variants to make a more informed choice for selection, they only take into account the lower bounds and require that the influence be calculated one additional time after optimization. This may still affect the accuracy but appears to make these methods slower so the overall effect is diminished.

• The difference between **local or global optimization** is also negligible (Fig. 8b). Global optimization gives slightly better accuracy, which shows that the time invested in optimizing all the leaves before Shannon expanding pays off more than investing in more frequent Shannon expansions.

Overall, the biggest factors are the (*i*) *choice of bounding method*, where the new optimization-based methods vastly outperform the existing ones, and (*ii*) the *number of optimization steps*, which should be kept small for our methods due to diminishing returns after the first optimization.

(3) Impact of data generation parameters. We next analyze the impact of dataset parameters on the performance for our new method (PGD$_d$) and the existing model-based method (MB) using their winning configurations (Fig. 7c).

Our optimization-based methods consistently outperform the prior methods for any query type, domain distribution, and lineage complexities on the synthetic data set.

• Figure 8d compares the impact of the **query type** (chain, cycle or star). On all three query types, our PGD$_d$ method outperforms the MB version; it is particularly effective on star queries, reducing the error to almost 0.

• Figure 8c shows the impact of the **domain distribution** used for drawing the probabilities. While exponentially distributed probabilities make the problem noticeably harder, the optimization-based method achieves consistently better accuracies than the model-based approach.

7.3 Accuracy Tradeoffs on Synthetic Data

To analyze the impact of lineage size and problem difficulty on the relative performance of PGD$_d$ and MB, we next (*i*) fix the 3-chain query, (*ii*) uniformly sample probabilities from $(0, 0.1]$, (*iii*) vary the number of tuples, and (*iv*) keep the domain size proportional to the number of tuples. We run each method for 2 minutes (or until it converges and thus performs exact probabilistic inference) and keep track of (1) the size of the lineage (x-axis) and (2) *when either method reaches a particular accuracy threshold* (y-axis).

Our optimization-based methods consistently outperform the prior methods for any lineage size and lineage complexities, at times by orders of magnitude.

Figure 9a shows the upper and lower bounds after the first evaluation of PGD$_d$. Figures 9b to 9d show the time / accuracy tradeoffs for different lineage sizes. Each dot represents one run achieving a certain accuracy threshold for the first time. We bin similarly sized lineages; if more than half the points

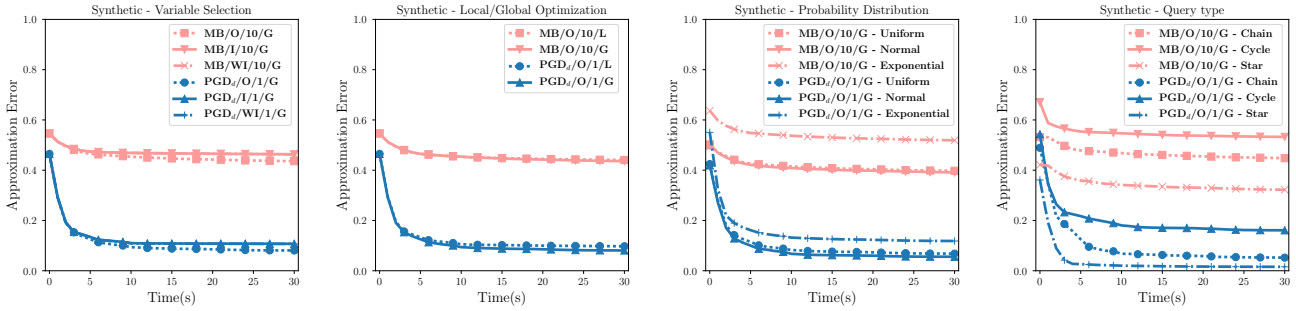


Figure 8: Section 7.2: Impact of variable selection strategy (a) and optimization strategy (b) are negligible. In contrast, query types (c) and domain distribution (d) affect the approximation error for PGD$_d$ and MB. Yet the advantage of our new methods is robust across all different regimes. Notice all plots show accuracy over time where a lower approximation error is better.

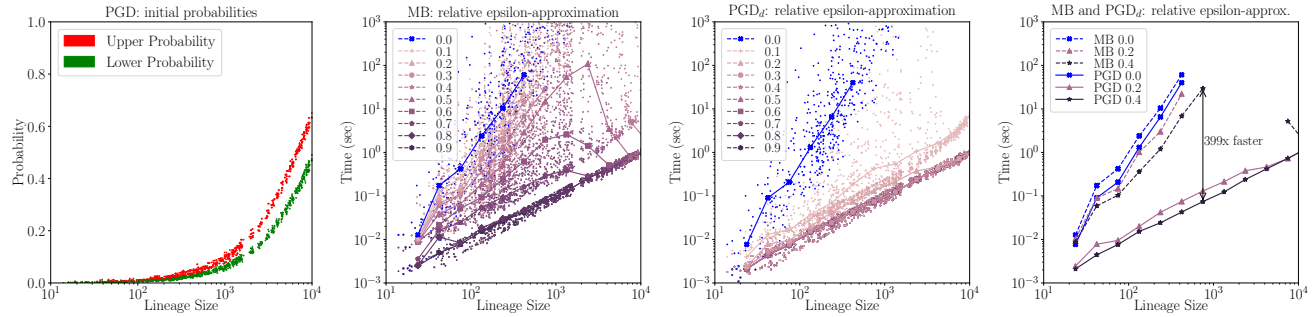


Figure 9: Section 7.3: Probability bounds (a) and *median* time to reach a certain approximation with either PGD$_d$ or MB (b)-(d). Our novel bounds allow us to reach the same accuracy by orders of magnitude faster than prior work (arrow illustrates 399×).

terminate, then we draw a summary point by taking the *median* assuming the timed out points at or above 2 minutes. The resulting contour lines then show the relative accuracy / time trade-offs between the methods over differently sized lineages. We see that for no regime does MB return the ε bound faster than PGD, whereas PGD is at times more than 2 orders of magnitude faster (MB times out, whereas PGD returns in less than 100 msec). For very large lineages (up to 200), the methods can still "converge", i.e., they decompose the lineage completely and thus perform exact inference. For very large lineages, when the probability becomes close to 1, then the lower bounds become high enough for all methods to give immediate good estimates.

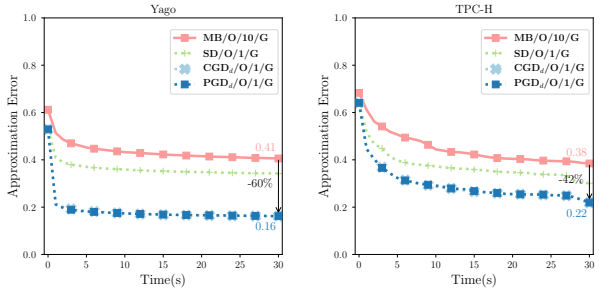


Figure 10: Section 7.4: Accuracy over time results over Yago3 and TPC-H, where a lower approximation error is better. We used the best configuration for each optimization method.

7.4 Yago & TPC-H

Yago. The Yago3 core dataset is originally an RDF dataset consisting of 55 million subject-predicate-object triples extracted from Wikipedia infoboxes. To turn the RDF dump of Yago3 into suitable input relations, over which self-join free queries can be posed, we horizontally partition the dataset into 79 relations based on the distinct predicates they contain. We examine four Boolean queries (Y1, Y2, Y3, Y4), where Y1 is an instantiation of the simplest template for an intractable query and consists of a three-way join between the relations `livesIn`, `worksAt` and `graduatedFrom`. All other queries are joins over the relations `isMarriedto`, `actedIn`, `directed` and `livesIn`, combined in different ways with `edited` and `hasWonPrize`. The precise queries are reported in the Appendix. To enlarge the class of queries, we additionally inject 3 different constants per query. Combined with the selections of a number of query plans and random seeds, as explained in Sect. 7.1, we report findings on 765 different lineages.

TPC-H. For the TPC-H dataset, we consider the Boolean versions of five queries (Q2B, Q9B, Q20B, Q21B, Q20WB)

from the original TPC-H benchmark, for which inference is hard. Here, Q20B is a join over `part`, `supplier`, `partsupplier`, and `nation`, to which Q2B adds `region`. Q21B is a join on `supplier`, `lineitem`, `orders` and `nation`, to which Q9B also adds `part` and `partsupplier`. These four queries[2] were also used in [34]. In addition, Q20WB is a variant of Q20B used in [20]. Just as for Yago3, we draw probabilities from a uniform, normal and exponential distribution, each initialized with 5 different seeds. We report findings on 975 different lineages.

The optimization-based methods ($\texttt{PGD}_d$ *and* $\texttt{CGD}_d$) *still outperform the model-based* (`MB`) *and symmetric dissociation-based* (`SD`) *methods (reduction of error by 42% and 60% against* `MB` *after 30 seconds, on average) on real datasets. In contrast to the synthetic experiments, the* `SD` *method performs here noticeably better than* (`MB`).

8 RELATED WORK

A plethora of approximation techniques for probabilistic query evaluation and inference have been developed over the past decades. Most related to our work are the following.

Sampling. Among the first approaches that introduced approximate confidence computations in probabilistic databases are the works by Ré and Suciu [37] in the context of the `MystiQ` [6, 40] system. Here, the authors employ a form of importance sampling based on the Monte-Carlo algorithm developed by Karp, Luby and Madras [28] for Boolean formulas in disjunctive normal form (DNF). Their work [40] also is the first PDB approach to develop a top-k pruning algorithm (coined "multi-simulation") based on lower and upper bounds for the marginal probabilities of the answer candidates. This line of works also led to the development of more principled criteria for efficient query answering in PDBs [8, 38] and the dichotomy theorem [9], which distinguishes among safe and non-safe query plans based on syntactic properties of the queries. We do not compare to sampling as `SPROUT` [16, 17], which is a special case of our method, was shown to outperform sampling-based methods.

Knowledge Compilation. Olteanu et al. [15, 32, 33] studied the application of knowledge-compilation techniques [11], first in the form of ordered binary decision diagrams (OBBDs) and later by own compilation techniques, in the context of the `MayBMS` [2] and `SPROUT` [34] systems. By focusing on OBDDs [32], their goal is to compile an entire lineage formula into a single OBDD, for which inference can then be performed in linear time in the size of the OBDD. This approach is not feasible for intractable queries, and we, by contrast, require incremental compilation of lineages.

Incremental Compilation. Our work is strongly inspired by the incremental compilation framework of [16, 17, 35]: it compiles lineages into d-trees and uses model-based approximations. While finding the best model-based bounds is a discrete optimization problem exponential in the number of variables, (*i*) our upper bounds are unique and provably better than any of the exponential many model-based bounds; (*ii*) our lower bounds form a continuum of bounds with model-based bounds as special cases at the ends of the continuum. *Embedding the problem into a continuous space allows us to use a variant of the work-horse for continuous optimization*: a variant of gradient-descent. This allows us to get better bounds before each decomposition step in practice, and to thus finish the incremental compilation earlier.

Dissociation. The model-based bounds of [16, 17, 35], which rewrite a Boolean formula into 1OF [44], are in fact a special case of the recently developed dissociation bounds [19–21]. Both can in turn be seen as a specific form of the Relax, Compensate, Recover framework by Choi and Darwiche [10], i.e., the idea to approximate a computationally hard problem by a relaxed instance of the problem, and then to compensate and recover the relaxed instance by incrementally moving back to the harder instance. As opposed to all existing works on both lineage-based and plan-level dissociations, we here investigate various non-uniform scaling approaches under given constraints.

Variable Elimination. Variable ordering in OBDDs [41], finally, is related to the considerations we take for variable elimination via Shannon expansions. For the latter, we however employ a form of sensitivity analysis [27] that has not been considered in [16, 17, 35] before. Moreover, as opposed to the static ordering considered in [41], our choice of the next variable to be eliminated is dynamically recomputed with every compilation step.

9 CONCLUSION

We propose an *anytime approximation framework* for probabilistic query evaluation that encompasses and generalizes prior state-of-the-art methods. We develop novel *scaled dissociation* bounds which are provably better than prior *model-based* (MB) or *symmetric dissociation* (SD) bounds, and show how they seamlessly integrate with an iterative compilation method. Through extensive experimentation, we confirm that our approach robustly outperforms prior work. While implemented for self-join conjunctive queries, our approach is more general as it relies only on Boolean expressions as input. Model-based approximation techniques are widely used in neighboring disciplines, such as statistical relational learning (SRL), weighted model counting (WMC), and general probabilistic inference. Further investigation is required to see how our framework can be applied in those areas too.

Acknowledgements This work was supported in part by grants NSF CAREER IIS-1762268 and FWO G042815N.

[2] http://www.cs.ox.ac.uk/dan.olteanu/papers/icde09queries.html

REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, pages 1479–1480, 2007.

[3] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.

[4] D. P. Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

[5] M. S. Boddy. Anytime problem solving using dynamic programming. In *AAAI*, pages 738–743, 1991.

[6] J. Boulos, N. N. Dalvi, B. Mandhani, S. Mathur, C. Ré, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.

[7] M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799, 2008.

[8] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[9] N. N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30:1–30:87, 2012.

[10] A. Darwiche. Relax, compensate and then recover: A theory of anytime, approximate inference. In *JELIA*, pages 7–9, 2010.

[11] A. Darwiche and P. Marquis. A knowledge compilation map. *J. Artif. Int. Res.*, 17(1):229–264, 2002.

[12] G. V. den Broeck and D. Suciu. Query processing on probabilistic data: A survey. *Foundations and Trends in Databases*, 7(3-4):197–341, 2017.

[13] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610, 2014.

[14] M. Dylla, I. Miliaraki, and M. Theobald. Top-k query processing in probabilistic databases with non-materialized views. In *ICDE*, pages 122–133, 2013.

[15] R. Fink, L. Han, and D. Olteanu. Aggregation in probabilistic databases via knowledge compilation. *PVLDB*, 5(5):490–501, 2012.

[16] R. Fink, J. Huang, and D. Olteanu. Anytime approximation in probabilistic databases. *VLDB J.*, 22(6):823–848, 2013.

[17] R. Fink and D. Olteanu. On the optimal approximation of queries using tractable propositional languages. In *ICDT*, pages 174–185, 2011.

[18] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[19] W. Gatterbauer and D. Suciu. Oblivious bounds on the probability of Boolean functions. *TODS*, 39(1):1–34, 2014.

[20] W. Gatterbauer and D. Suciu. Approximate lifted inference with probabilistic databases. *PVLDB*, 8(5):629–640, 2015.

[21] W. Gatterbauer and D. Suciu. Dissociation and propagation for approximate lifted inference with standard relational database management systems. *VLDB J.*, 26(1):5–30, 2017.

[22] V. Gogate and P. M. Domingos. Probabilistic theorem proving. In *UAI*, pages 256–265, 2011.

[23] M. C. Golumbic and V. Gurvich. *Read-once functions*, chapter 10. Boolean Functions: Theory, Algorithms and Applications. Cambridge University Press, 2010.

[24] C. P. Gomes, A. Sabharwal, and B. Selman. Model counting. In *Handbook of Satisfiability*, pages 633–654. IOS Press, 2009.

[25] V. Gurvich. Repetition-free Boolean functions. *Uspekhi Mat. Nauk*, 32:183–184, 1977. (in Russian).

[26] A. K. Jha and D. Suciu. Probabilistic databases with markoviews. *PVLDB*, 5(11):1160–1171, 2012.

[27] B. Kanagal, J. Li, and A. Deshpande. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *SIGMOD*, pages 841–852, 2011.

[28] R. M. Karp, M. Luby, and N. Madras. Monte-carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.

[29] N. Khoussainova, M. Balazinska, and D. Suciu. Probabilistic event extraction from RFID data. In *ICDE*, pages 1480–1482, 2008.

[30] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[31] F. Mahdisoltani, J. Biega, and F. M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR*, 2015.

[32] D. Olteanu and J. Huang. Using OBDDs for efficient query evaluation on probabilistic databases. In *SUM*, pages 326–340, 2008.

[33] D. Olteanu and J. Huang. Secondary-storage confidence computation for conjunctive queries with inequalities. In *SIGMOD*, pages 389–402, 2009.

[34] D. Olteanu, J. Huang, and C. Koch. SPROUT: lazy vs. eager query plans for tuple-independent probabilistic databases. In *ICDE*, pages 640–651, 2009.

[35] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *ICDE*, pages 145–156, 2010.

[36] L. D. Raedt and K. Kersting. Statistical relational learning. In *Encyclopedia of Machine Learning and Data Mining*, pages 1177–1187. Springer, 2017.

[37] C. Ré, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, pages 886–895, 2007.

[38] C. Ré, N. N. Dalvi, and D. Suciu. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull.*, 29(1):25–31, 2006.

[39] C. Ré and D. Suciu. Approximate lineage for probabilistic databases. *PVLDB*, 1(1):797–808, 2008.

[40] C. Ré and D. Suciu. Managing probabilistic data with MystiQ: The can-do, the could-do, and the can't-do. In *SUM*, pages 5–18, 2008.

[41] M. N. Rice and S. Kulhari. A survey of static variable ordering heuristics for efficient BDD/MDD construction. In *University of California, Tech. Rep.*, 2008.

[42] T. Sang, P. Bearne, and H. Kautz. Performing Bayesian inference by weighted model counting. In *AAAI*, pages 475–481, 2005.

[43] P. Sen, A. Deshpande, and L. Getoor. Exploiting shared correlations in probabilistic databases. *PVLDB*, 1(1):809–820, 2008.

[44] P. Sen, A. Deshpande, and L. Getoor. Read-once functions and query evaluation in probabilistic databases. *PVLDB*, 3(1):1068–1079, 2010.

[45] J. Shin, S. Wu, F. Wang, C. De Sa, C. Zhang, and C. Ré. Incremental knowledge base construction using DeepDive. *PVLDB*, 8(11):1310–1321, 2015.

[46] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool, 2011.

[47] TPC-H Benchmark. http://www.tpc.org/tpch/.

[48] W. Wang and M. Á. Carreira-Perpiñán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *CoRR*, abs/1309.1541, 2013.

[49] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492, 2012.

[50] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

A IMPLEMENTATION ISSUES

For both gradient methods, we need to set step sizes (η_t) and compute the gradient in each step.

Step size. The correctness of gradient methods relies on selecting appropriate step sizes. That is, step sizes should be selected, such that (i) $G_{\varphi'}(\boldsymbol{\alpha}^{(t+1)}) < G_{\varphi'}(\boldsymbol{\alpha}^{(t)})$, and (ii) limit points of the sequence $\boldsymbol{\alpha}^{(t)}$ are so-called *stationary points* $\boldsymbol{\alpha}^\star$, i.e., points for which $\nabla G_{\varphi'}(\boldsymbol{\alpha}^\star)' \cdot (\boldsymbol{\alpha} - \boldsymbol{\alpha}^\star) \geq 0$, for every $\boldsymbol{\alpha} \in \Delta$ (see [4] for more details). This is the best one can hope for, since (local) minima of $G_{\varphi'}$ are necessarily found at these stationary points. It is only when, e.g., $G_{\varphi'}$ is convex that a stationary point is guaranteed to result in the global minimum value.

Various methods exist for finding appropriate step sizes, e.g., limited minimization, Armijo rules, and other line-search methods [4]. These methods all rely on the evaluation of $G_{\varphi'}$ in multiple points along the gradient direction, to ensure that conditions (i) and (ii) are met. As observed earlier, although evaluating $G_{\varphi'}$ is in PTIME, the time complexity depends on the size of the 1OF lineage φ'. In realistic settings, the size of φ' can be large (Fig. 5) and the evaluation of $G_{\varphi'}$ causes a substantial overhead.

We can show, however, that there exists a constant $L > 0$ (called a *Lipschitz* constant), such that for any $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \Delta$, the inequality $\|\nabla G_{\varphi'}(\boldsymbol{\alpha}) - \nabla G_{\varphi'}(\boldsymbol{\beta})\|_2 \leq L\|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2$ holds. Furthermore, the Lipschitz constant L only depends on the original probabilities of the variables in φ and on the number of their occurrences d_i and can be easily computed (see Appendix B). This implies that we can choose step sizes as follows:

- **constant step size** $0 < \eta_t \leq 2/L$ for PGD (Prop. 3.3.2 in [4])
- **diminishing step size** $\eta_t := \min\{1, \frac{\nabla G_{\varphi'}(\boldsymbol{\alpha}^{(t)})' \cdot (\boldsymbol{\gamma}^{(t)} - \boldsymbol{\alpha}^{(t)})}{2nL}\}$ for CGD (Ex. 3.2.5 in [4])

It is then *guaranteed* that convergence points of PGD and CGD are indeed stationary points. More importantly, by setting η_t as described above, we *avoid* costly computations of $G_{\varphi'}$ at intermediate points along the gradient direction in each step of the methods. This is *crucial* for the application of gradient methods in the overall anytime approximation algorithm.

Gradient computation. The gradient $\nabla G_{\varphi'}$ of $G_{\varphi'}$ consists of the partial derivatives $\frac{\partial G_{\varphi'}}{\partial \alpha_{ij}}(\cdot)$, which in turn are equal to $\ln(\bar{p}(x_i))(1 - x'_{ij})^{\alpha_{ij}}\, infl_{x'_{ij},\varphi'}(\cdot)$. This is immediate from applying the chain rule of derivatives and our form of variable substitution. This close relationship to the *influence* of variables (Sect. 2), and the fact that φ' is in 1OF implies that the gradient can be efficiently computed [27].

B EXISTENCE OF LIPSCHITZ CONSTANT

We show that the objective function $G_{\varphi'} : \mathbb{R}^n \to \mathbb{R}$ in the optimization problem (SLB′) in Sect. 4 satisfies the Lipschitz condition. More precisely, we show the following theorem.

Theorem 2. *Let φ' be a dissociation of φ obtained by dissociation the n variables $x_1, \ldots, x_n$ in φ. Let*

$$L := \max_{i \in [n]} \Big\{ \ln(\bar{p}_i)^2 + (d_i - 1)\ln(\bar{p}_i)^2 \bar{p}_i^{1/d_i} + \sum_{\substack{j=1 \\ j \neq i}}^{n} d_j \ln(\bar{p}_i)\ln(\bar{p}_j)\bar{p}_j^{1/d_j} \Big\},$$

where $d_i = |\theta^{-1}(x_i)|$ for θ the substitution such that $\varphi'[\theta] = \varphi$, and p_i denotes the probability of x_i, for $i \in [n]$. Then, for any $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \Delta$,

$$\|\nabla G_{\varphi'}(\boldsymbol{\alpha}) - \nabla G_{\varphi'}(\boldsymbol{\beta})\|_2 \leq L\|\boldsymbol{\alpha} - \boldsymbol{\beta}\|_2 \tag{2}$$

The theorem is shown by bounding the spectral radius $\rho\big(\mathbf{J}_{G_{\varphi'}}(\boldsymbol{\alpha}, \boldsymbol{\beta})\big)$ of the Jacobian of $G_{\varphi'}$ by the value L defined in the statement of the Theorem. We recall that the spectral radius is the largest (in absolute value) eigenvalue of the Jacobian. It is known that this suffices for showing the inequality (2).

Proof. (Sketch) Recall that $G_{\varphi'}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is defined in terms of function $F_{\varphi'}(\mathbf{x}, \mathbf{y}) = \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi')$. It is known (see [27, 39]) that $F_{\varphi'}(\mathbf{x}, \mathbf{y})$ is a multi-linear function and furthermore that $\frac{\partial F_{\varphi'}}{\partial z}(\mathbf{x}, \mathbf{y}) = \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[z/1]) - \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[z/0])$ for each variable $z \in \mathbf{x} \cup \mathbf{y}$. Since in our setting, φ' is a positive formula, i.e., each literal appears positive (i.e., not negated), one can verify that $0 \leq \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[z/0]) \leq \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[z/1])$ and hence $\frac{\partial F_{\varphi'}}{\partial z}(\mathbf{x}, \mathbf{y}) \in [0, 1]$. Since we are working with $G_{\varphi'}$ rather than $F_{\varphi'}$, we first relate the partial derivatives of $G_{\varphi'}$ to those of $F_{\varphi'}$. Using the chain rule for derivatives, we obtain: $\frac{\partial G_{\varphi'}}{\partial \alpha_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) := \ln(\bar{p})\bar{p}^{\alpha_i} \frac{\partial F_{\varphi'}}{\partial x_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$, for $i \in [d]$, and $\frac{\partial G_{\varphi'}}{\partial \beta_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) := \ln(\bar{q})\bar{q}^{\beta_i} \frac{\partial F_{\varphi'}}{\partial y_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$, for $i \in [e]$. Since $\ln(\bar{p}) \leq 0$ and $\ln(\bar{q}) \leq 0$, our earlier observation implies that $\ln(\bar{p})\bar{p}^{\alpha_i} \leq \frac{\partial G_{\varphi'}}{\partial \alpha_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leq 0$ and $\ln(\bar{q})\bar{q}^{\beta_i} \leq \frac{\partial G_{\varphi'}}{\partial \beta_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leq 0$. We next inspect the elements of the Jacobian of $G_{\varphi'}$ by computing its second order derivatives. It is readily verified that: $a_{ii} := \frac{\partial^2 G_{\varphi'}}{\partial \alpha_i^2}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -(\ln(\bar{p}))^2 p^{2\alpha_i} \frac{\partial^2 F_{\varphi'}}{\partial x_i^2}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}}) + (\ln(\bar{p}))^2 \bar{p}^{\alpha_i} \frac{\partial F_{\varphi'}}{\partial x_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$ which is equal to $(\ln(\bar{p}))^2 \bar{p}^{\alpha_i} \frac{\partial F_{\varphi'}}{\partial x_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$, where we used that $\frac{\partial^2 F_{\varphi'}}{\partial x_i^2}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}}) = 0$ (multi-linearity of $F_{\varphi'}$). Similarly, $b_{ii} := \frac{\partial^2 G_{\varphi'}}{\partial \beta_i^2}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\ln(\bar{q}))^2 \bar{p}^{\beta_i} \frac{\partial F_{\varphi'}}{\partial y_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$ and for $i \neq j$, $a_{ij} := \frac{\partial^2 G_{\varphi'}}{\partial \alpha_j \partial \alpha_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -(\ln(\bar{p}))^2 \bar{p}^{\alpha_i + \alpha_j} \frac{\partial^2 F_{\varphi'}}{\partial x_j \partial x_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$, $b_{ij} := \frac{\partial^2 G_{\varphi'}}{\partial \beta_j \partial \beta_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -(\ln(\bar{q}))^2 \bar{q}^{\beta_i + \beta_j} \frac{\partial^2 F_{\varphi'}}{\partial y_j \partial y_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$, and $c_{ij} := \frac{\partial^2 G_{\varphi'}}{\partial \beta_j \partial \alpha_i}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\ln(\bar{p})\ln(\bar{q})\bar{p}^{\alpha_i}\bar{q}^{\beta_j} \frac{\partial^2 F_{\varphi'}}{\partial y_j \partial x_i}(1 - \bar{p}^{\boldsymbol{\alpha}}, 1 - \bar{q}^{\boldsymbol{\beta}})$. We will bound the spectral radius of $\mathbf{J}_{G_{\varphi'}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ by means of Gershgorin's circle theorem. This theorem tells

us that for each eigenvalue λ of $\mathbf{J}_{G_{\varphi'}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$, there exists a row i such that when $1 \leq i \leq d$, $|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{d} |a_{ij}| + \sum_{j=1}^{e} |c_{ij}|$, and when $1 \leq i \leq e$: $|\lambda - b_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{e} |b_{ij}| + \sum_{j=1}^{d} |c_{ij}|$. It remains to bound the entries a_{ii}, a_{ij}, b_{ii}, b_{ij} and c_{ij} in $\mathbf{J}_{G_{\varphi'}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. We also point out that since the Jacobian is a symmetric matrix all these eigenvalues are real. From our earlier observation on the derivatives $\frac{\partial F_{\varphi'}}{\partial x_i}$ and $\frac{\partial F_{\varphi'}}{\partial y_i}$ it easily follows that $a_{ii} \in [0, (\ln(\bar{p})^2 \bar{p}^{\alpha_i}]$ and $b_{ii} \in [0, (\ln(\bar{q})^2 \bar{q}^{\beta_i}]$. Furthermore, for any $\mathbf{x} \in [0,1]^d$ and $\mathbf{y} \in [0,1]^e$ it is easily verified that

$$\left|\frac{\partial^2 F_{\varphi'}}{\partial x_j \partial x_i}(\mathbf{x}, \mathbf{y})\right| \quad , \left|\frac{\partial^2 F_{\varphi'}}{\partial y_j \partial x_i}(\mathbf{x}, \mathbf{y})\right| \text{ and } \left|\frac{\partial^2 F_{\varphi'}}{\partial y_j \partial y_i}(\mathbf{x}, \mathbf{y})\right|$$

are all bounded by 1. Indeed, this following from the observation that, e.g., $\frac{\partial^2 F_{\varphi'}}{\partial x_j \partial x_i}(\mathbf{x}, \mathbf{y})$ is given by $(\mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/1, x_j/1]) - \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/1, x_j/0])) - (\mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/0, x_j/1]) - \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/0, x_j/0]))$ where each of the terms take values in $[0,1]$ and furthermore, $(\mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/1, x_j/1]) - \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/1, x_j/0])) \geq 0$ and $(\mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/0, x_j/1]) - \mathbb{P}_{\mathbf{x},\mathbf{y}}(\varphi'[x_i/0, x_j/0]))$ hold because φ' is positive. As a consequence, $|a_{ij}| \leq (\ln(\bar{p}))^2 \bar{p}^{\alpha_i + \alpha_j}$, $|b_{ij}| \leq (\ln(\bar{q}))^2 \bar{q}^{\beta_i + \beta_j}$, and $|c_{ij}| \leq \ln(\bar{p}) \ln(\bar{q}) \bar{p}^{\alpha_i} \bar{q}^{\beta_j}$. Hence, each eigenvalue λ satisfies either $|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{d} (\ln(\bar{p}))^2 \bar{p}^{\alpha_i + \alpha_j} + \ln(\bar{p}) \ln(\bar{q}) \bar{p}^{\alpha_i} \left(\sum_{j=1}^{e} \bar{q}^{\beta_j}\right)$, for some $i \in [1,d]$, or $|\lambda - b_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{e} (\ln(\bar{q}))^2 \bar{q}^{\beta_i + \beta_j} + \ln(\bar{q}) \ln(\bar{p}) \bar{q}^{\beta_i} \left(\sum_{j=1}^{d} \bar{p}^{\alpha_j}\right)$, for some $i \in [1,e]$. We continue bounding the right-hand sides of these inequalities. Consider first the sum $s(\boldsymbol{\alpha}) = \sum_{j=1}^{d} \bar{p}^{\alpha_j}$. Following Example 3.1.2 in [4], $s(\boldsymbol{\alpha})$ reaches a maximal value in Δ_d at a point $\boldsymbol{\alpha}^\star$ in which all non-zero entries have the same partial derivative value and furthermore, the partial derivative at zero entries are smaller than that value. Observe that $\frac{\partial s}{\partial \alpha_j}(\boldsymbol{\alpha}^\star) = \ln(\bar{p}) \bar{p}^{\alpha_j^\star}$ and hence to have equal derivatives, the non-zero entries in $\boldsymbol{\alpha}^\star$ must be equal. Suppose that we have k non-zero entries in $\boldsymbol{\alpha}^\star$, each of which equal to $1/k$. All remaining $d-k$ entries are zero, in which the partial derivative is $\ln(\bar{p})$ which is smaller than $\ln(\bar{p}) \bar{p}^{1/k}$ since $\ln(\bar{p})$ is negative. As a consequence, it suffices to check which of the candidate points $\boldsymbol{\alpha}^\star$ maximises s. This in turn pours down to checking how many non-zero entries $\boldsymbol{\alpha}^\star$ must have to maximise s. Observe now that s takes value $k \bar{p}^{1/k}$ at such points. Since $1/k > 1/(k+1)$ implies $\ln(\bar{p}) 1/k < \ln(\bar{p}) 1/(k+1)$ and hence $\bar{p}^{1/k} < \bar{p}^{1/(k+1)}$, we have that s reaches a maximum when $k = d$. This implies that $a_i^\star = 1/d$ for all $i \in [d]$. A similar reasoning shows that $\sum_{j=1}^{e} \ln(\bar{q}) \bar{q}^{\beta_j}$ reaches it maximal value in Δ_e at a point $\boldsymbol{\beta}^\star$ such that $\beta_i^\star = 1/e$ for all $i \in [e]$. Hence, we thus have that each eigenvalue λ satisfies either $|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{d} (\ln(\bar{p}))^2 \bar{p}^{\alpha_i + \alpha_j} + \ln(\bar{p}) \ln(\bar{q}) \bar{p}^{\alpha_i} e \bar{q}^{1/e}$ for some $i \in [1,d]$, or $|\lambda - b_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{e} (\ln(\bar{q}))^2 \bar{q}^{\beta_i + \beta_j} + \ln(\bar{p}) \ln(\bar{q}) \bar{q}^{\beta_i} d \bar{p}^{1/d}$, for some $i \in [1,e]$. Now, in these right-hand side expressions in the inequalities we have only one type of variable left, i.e., either variables in $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$. We next find $\boldsymbol{\alpha}^\star$ (resp. $\boldsymbol{\beta}^\star$) that maximise these expressions. Let us use the shorthand notation $t(\boldsymbol{\alpha})$ for $\sum_{\substack{j=1 \\ j \neq i}}^{d} (\ln(\bar{p}))^2 p^{\alpha_i + \alpha_j} + \ln(\bar{p}) \ln(\bar{q}) e \bar{q}^{1/e} \bar{p}^{\alpha_i}$. Again, following Example 3.1.2 in [4], we identify conditions on $\boldsymbol{\alpha}^\star$. The partial derivatives of $t(\boldsymbol{\alpha})$ are non-positive and given by $\frac{\partial t}{\partial \alpha_i}(\boldsymbol{\alpha}) = (d-1)(\ln \bar{p})^3 \bar{p}^{\alpha_i + \alpha_j} + \ln(\bar{p})^2 \ln(\bar{q}) e \bar{q}^{1/e} \bar{p}^{\alpha_i}$ and $\frac{\partial t}{\partial \alpha_j}(\boldsymbol{\alpha}) = (\ln \bar{p})^3 \bar{p}^{\alpha_i + \alpha_j}$. Observe that $\frac{\partial t}{\partial \alpha_i}(\boldsymbol{\alpha}) < \frac{\partial t}{\partial \alpha_j}(\boldsymbol{\alpha})$ for all $j \neq i$. Hence, in $\boldsymbol{\alpha}^\star$ we must have that $\alpha_i^\star = 0$ and at such points t is of the form $\sum_{\substack{j=1 \\ j \neq i}}^{d} (\ln(\bar{p}))^2 p^{\alpha_j} + \ln(\bar{p}) \ln(\bar{q}) e \bar{q}^{1/e}$. In a similar way as above, it suffices to maximise $\sum_{\substack{j=1 \\ j \neq i}}^{d} p^{\alpha_j}$, which as argued earlier is obtained when each $\alpha_j^\star = 1/(d)$ for $j \neq i$. The same holds for $\sum_{\substack{j=1 \\ j \neq i}}^{e} (\ln(\bar{q}))^2 \bar{q}^{\beta_i + \beta_j} + \ln(\bar{p}) \ln(\bar{q}) \bar{q}^{\beta_i} d \bar{p}^{1/d}$, where now $\beta_j^\star = 1/e$ for $j \neq i$. So, to summarize, we have that each eigenvalue λ satisfies either $|\lambda - a_{ii}| \leq (d-1) \ln(\bar{p})^2 \bar{p}^{1/d} + e \ln(\bar{p}) \ln(\bar{q}) \bar{q}^{1/e}$ for some $i \in [1,d]$, or $|\lambda - b_{ii}| \leq (e-1) \ln(\bar{q})^2 \bar{q}^{1/e} + d \ln(\bar{q}) \ln(\bar{p}) \bar{p}^{1/d}$, for some $i \in [1,e]$. Combined with the bounds on a_{ii} and b_{ii}, we may conclude that each eigenvalue λ of $\mathbf{J}_{G_{\varphi'}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ satisfies either $|\lambda| \leq \ln(\bar{p})^2 + (d-1) \ln(\bar{p})^2 \bar{p}^{1/d} + e \ln(\bar{p}) \ln(\bar{q}) \bar{q}^{1/e}$ or $|\lambda| \leq \ln(\bar{q})^2 + (e-1) \ln(\bar{q})^2 \bar{q}^{1/e} + d \ln(\bar{q}) \ln(\bar{p}) \bar{p}^{1/d}$. We may thus define L as the maximum of these two values.

In general, when φ' is a dissociation of φ in which n variables $x_1, \ldots, x_n$ are dissociated, let $d_i = |\theta^{-1}(x_i)|$ where θ is the substitution such that $\varphi'[\theta] = \varphi$. If p_i denotes the probability of x_i, for $i \in [n]$, then the previous reasoning (for $n = 2$) results in a Lipschitz constant $L := \max_{i \in [n]} \{\ln(\bar{p}_i)^2 + (d_i - 1) \ln(\bar{p}_i)^2 \bar{p}_i^{1/d_i} + \sum_{\substack{j=1 \\ j \neq i}}^{n} d_j \ln(\bar{p}_i) \ln(\bar{p}_j) \bar{p}_j^{1/d_j}\}$. □

C CORRECTNESS OF AAA

We verify the correctness of algorithm AAA (2 in Sect. 5).

Proposition 2. *After each step of the algorithm* AAA*, every leaf ψ in the d-tree has a 1OF-dissociation ψ' and probability functions p_L and p_U associated with it, such that $\mathbb{P}_L[\psi'] \leq \mathbb{P}[\psi] \leq \mathbb{P}_U[\psi']$ holds. Furthermore, the lower and upper bounds* never get worse *at each step of the algorithm.*

Proof. Recall that algorithm AAA starts from a lineage φ and 1OF dissociation φ' (for substitution θ) of φ. In addition, lower and upper bound probability functions p_L and p_U are provided such that $\mathbb{P}_{p_L}(\varphi') \leq \mathbb{P}_p(\varphi) \leq \mathbb{P}_{p_U}(\varphi')$ holds. During a run of the algorithm φ gets decomposed in a d-tree, and furthermore, φ' gets decomposed along the way in precisely

the same way. Assume that AAA has run for a number of iterations and assume that when looking at each leaf ψ in the current d-tree φ, there is a corresponding 1OF dissociation ψ' of ψ and probability functions p_L and p_U for the variables in ψ' such that $\mathbb{P}_{p_L}(\psi') \leq \mathbb{P}_p(\psi) \leq \mathbb{P}_{p_U}(\psi')$ is satisfied. We need to show that at the end of the current iteration of the algorithm, the updated d-trees φ, φ' and probability functions p_L and p_U at the leaves still satisfy $\mathbb{P}_{p_L}(\psi') \leq \mathbb{P}_p(\psi) \leq \mathbb{P}_{p_U}(\psi')$, and this for every leaf ψ' (in φ') and ψ (in φ). We also recall that, as described in section 5, that when we consider P_L and P_U as the collection of all functions p_L and p_U, respectively, in the leaves of φ', we have $\mathbb{P}_{P_L}(\varphi') \leq \mathbb{P}_p(\varphi) \leq \mathbb{P}_{P_U}(\varphi')$. We also show that the bounds never get worse after an iteration of AAA. For starters, let us explore how procedure OptimizeBounds impact the d-trees and functions under consideration. As described in Sect. 5 this procedure simply updates the probability functions p_L (into p'_L) and p_U (into p'_U) at the leaves of the current d-tree, and the proposed update is only accepted when it does not revise the current bounds, $\mathbb{P}_{p_L}(\psi') \leq \mathbb{P}_p(\psi) \leq \mathbb{P}_{p_U}(\psi')$, for the worse. Clearly, this implies that $\mathbb{P}_{p_L}(\psi') \leq \mathbb{P}_{p'_L}(\psi')$ and $\mathbb{P}_{p'_U}(\psi') \leq \mathbb{P}_{p_U}(\psi')$ for all leaves. Furthermore, since OptimizeBounds is expected to update probability functions in accordance to the conditions stated in Theorem 1, and the procedure does not change the d-trees (only probability functions associated with leaves are changed), $\mathbb{P}_{p'_L}(\psi) \leq \mathbb{P}_p(\psi) \leq \mathbb{P}_{p'_U}(\psi')$ is still satisfied after the call to procedure OptimizeBounds. So, after the call to OptimizeBounds all our desired properties remain satisfied.

It remains to inspect the impact of decompositions ($\otimes$ or $\odot$) and Shannon expansions ($\oplus$). We consider each leaf ψ and corresponding 1OF dissociation ψ' in isolation.

- Independent "or" ($\otimes$): Here, $\psi = \psi_1 \vee \psi_2$ with $vars(\psi_1) \cap vars(\psi_2) = \emptyset$. The same holds for its 1OF dissociation, i.e., $\psi' = \psi'_1 \vee \psi'_2$ with $vars(\psi'_1) \cap vars(\psi'_2) = \emptyset$. By considering, for $i = 1, 2$, the substitution $\theta_i = \theta|_{\theta^{-1}(vars(\psi_i))}$, the restriction of θ to the variables in ψ' that map to variables in ψ_i, is such that $\psi'_i[\theta_i] = \psi_i$. Hence, for $i = 1, 2$, ψ'_i is a 1OF dissociation of ψ_i. Furthermore, by defining p^i_L and p^i_U as the probability functions obtained by restricting p_L and p_U, respectively, to the variables in ψ'_i, it easily follows that the conditions of Theorem 1 remain satisfied, i.e., $\mathbb{P}_{p^i_L}(\psi'_i) \leq \mathbb{P}[\psi_i] \leq \mathbb{P}_{p^i_U}(\psi'_i)$.
- Independent "and" ($\odot$): This case is completely analogous to the previous case, but using $\psi = \psi_1 \wedge \psi_2$ with $vars(\psi_1) \cap vars(\psi_2) = \emptyset$ and dissociations $\psi' = \psi'_1 \wedge \psi'_2$ with $vars(\psi'_1) \cap vars(\psi'_2) = \emptyset$ instead.

So, 2 holds after $\otimes$ and $\odot$ decomposition. Furthermore, $\mathbb{P}_{p_L}(\psi') = \mathbb{P}_{p^1_L}(\psi'_1) \otimes \mathbb{P}_{p^2_L}(\psi'_2)$ and $\mathbb{P}_{p_U}(\psi') = \mathbb{P}_{p^1_U}(\psi'_1) \otimes \mathbb{P}_{p^2_U}(\psi'_2)$ in case of an independent "or" decomposition, and similarly, $\mathbb{P}_{p_L}(\psi') = \mathbb{P}_{p^1_L}(\psi'_1) \cdot \mathbb{P}_{p^2_L}(\psi'_2)$ and $\mathbb{P}_{p_U}(\psi') = \mathbb{P}_{p^1_U}(\psi'_1) \cdot \mathbb{P}_{p^2_U}(\psi'_2)$ in case of an independent "and" decomposition. In other words, the lower and upper bounds remain the same. Lifting this to the lineages φ and φ', the probability of $\mathbb{P}_{P_L}(\varphi')$ and $\mathbb{P}_{P_U}(\varphi')$ are unchanged after such decompositions. We next explore what happens after a Shannon expansion.

- Shannon expansions ($\oplus$): Let us assume that a leaf ψ in d-tree φ and variable $x \in vars(\psi)$ is selected. Then, ψ gets decomposed as $(x \odot \psi_1) \oplus (\neg x \odot \psi_2)$ with $\psi_1 = \psi[1/x]$ and $\psi_2 = \psi[0/x]$. We have $\mathbb{P}_p(\psi) = p(x) \cdot \mathbb{P}_p(\psi_1) + \bar{p}(x) \cdot \mathbb{P}_p(\psi_2)$. We let ψ'_1 be the 1OF formula obtained from ψ' by setting all variables x_i, such that $\theta(x_i) = x$, to true; similarly, ψ'_2 is the 1OF formula obtained from ψ' by setting all variables x_i, such that $\theta(x_i) = x$, to false. We let θ' be the restriction of θ to those variables in ψ that do not map to x. Clearly, $\psi'_1[\theta'] = \psi_1$ and $\psi'_2[\theta'] = \psi_2$. Recall that we assume that $vars(\psi'_1) \cap vars(\psi'_2) = \emptyset$. We can always ensure this by applying a variable renaming. Let ψ'_2 also denote the formula (after this possible renaming), then clearly, there is a substitution $\theta_2 = \theta' \circ \rho$ (with ρ being the renaming) such that $\psi'_2[\theta_2] = \psi_2$. We refer to θ' as θ_1 such that $\psi'_1[\theta_1] = \psi_1$. For p_L and p_U, we decompose them as expected, i.e., restrict them to the relevant variables (taking the renaming ρ into account). It is again easily verified that $\mathbb{P}_{p^1_L}(\psi'_1) \leq \mathbb{P}[\psi_1] \leq \mathbb{P}_{p^1_U}(\psi'_1)$ and $\mathbb{P}_{p^2_L}(\psi'_2) \leq \mathbb{P}[\psi_2] \leq \mathbb{P}_{p^1_U}(\psi'_2)$, because the condition of Theorem 1 are satisfied. So, 2 holds after Shannon expansion.

We still need to show that after Shannon expansion the bounds did not deteriorate. We already known from the previous analysis that $p(x) \cdot \mathbb{P}_{p^1_L}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_L}(\psi'_2) \leq \mathbb{P}[\psi]$ and similarly $\mathbb{P}[\psi] \leq p(x) \cdot \mathbb{P}_{p^1_U}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_U}(\psi'_2)$. It remains to show that $\mathbb{P}_{p_L}(\psi') \leq p(x) \cdot \mathbb{P}_{p^1_L}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_L}(\psi'_2)$ and $p(x) \cdot \mathbb{P}_{p^1_U}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_U}(\psi'_2) \leq \mathbb{P}_{p_U}(\psi')$ hold. Indeed, if these hold then $\mathbb{P}_{p_L}(\psi') \leq p(x) \cdot \mathbb{P}_{p^1_L}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_L}(\psi'_2) \leq \mathbb{P}[\psi]$, and similarly, $\mathbb{P}[\psi] \leq p(x) \cdot \mathbb{P}_{p^1_U}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_U}(\psi'_2) \leq \mathbb{P}_{p_U}(\psi')$. In other words, the bounds do not get worse. Observe that $p(x) \cdot \mathbb{P}_{p^1_L}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_L}(\psi'_2)$ and $p(x) \cdot \mathbb{P}_{p^1_U}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_U}(\psi'_2)$ can be seen as the probability of a Boolean formula ξ, for which ψ' is a dissociation to which Theorem 1 applies. From this, the inequalities then follow. The formula ξ is simply $\psi'[x/x']$, the formula ψ' in which each dissociated variable $x' \in \theta^{-1}(x)$ is replaced by x. It is easily verified that for probability function p'_L such that $p'_L(x) = p(x)$, $p'_L(y) = p^1_L(y)$ for variables y in ψ'_1, and $p'_L(y) = p^2_L(y)$ for variables y in ψ'_2, and p'_U such that $p'_U(x) = p(x)$, $p'_U(y) = p^1_U(y)$ for variables y in ψ'_1, and $p'_U(y) = p^2_U(y)$ for variables y in ψ'_2, we have that $\mathbb{P}_{p'_L}(\xi) = p(x) \cdot \mathbb{P}_{p^1_L}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_L}(\psi'_2)$ and $\mathbb{P}_{p'_U}(\xi) = p(x) \cdot \mathbb{P}_{p^1_U}(\psi'_1) + \bar{p}(x) \cdot \mathbb{P}_{p^2_U}(\psi'_2)$, as desired. We can now apply Theorem 1 for dissociation ψ' of ξ and the probability functions defined above. This results in $\mathbb{P}_{p_L}(\psi') \leq \mathbb{P}_{p'_L}(\xi)$ and $\mathbb{P}_{p_U}(\psi') \geq \mathbb{P}_{p'_U}(\xi)$, from which the desired inequalities follow. Hence, also after a Shannon expansion the bounds never get worse. □