General suggestion: Do NOT define BIDB / TIDB in possible world semantics. I think trying to fit stuff into possible world semantics just make notation more complicated

equivalent SMB representation of Φ (without materializing the SMB representation) and 'adjust' their contribution to $\widetilde{\Phi}(\cdot)$.

Applications. Recent work in heuristic data cleaning [49, 43, 40, 8, 43] emits a PDB when insufficient data exists to select the 'correct' data repair. Probabilistic data cleaning is a crucial innovation, as the alternative is to arbitrarily select one repair and 'hope' that queries receive meaningful results. Although PDB queries instead convey the trustworthiness of results [35], they are impractically slow [18, 17], even in approximation (see Appendix G). Bags, as we consider, are sufficient for production use, where bag-relational algebra is already the default for performance reasons. Our results show that bag-PDBs can be competitive, laying the groundwork for probabilistic functionality in production database engines.

Or use $X_1, \ldots, X_m$ for generic m

Paper Organization. We present relevant background and notation in Sec. 2. We then prove our main hardness results in Sec. 3 and present our approximation algorithm in Sec. 4. Finally, we discuss related work in Sec. 5 and conclude in Sec. 6. All proofs are in the appendix.

For Sec 2.1 perhaps use a generic set of variables S & define everything in terms of that?

2 Background and Notation

2.1 Polynomial Definition and Terminology

A polynomial over $\mathbf{X} = (X_1, \ldots, X_n)$ with individual degree $B < \infty$ is formally defined as (where $c_{\mathbf{d}} \in \mathbb{N}$):

not consistent.

$$\Phi(X_1, \ldots, X_n) = \sum_{\mathbf{d}\in\{0,\ldots,B\}^D} c_{\mathbf{d}} \cdot \prod_{t\in D} X_t^{d_t}. \tag{1}$$

▶ **Definition 2.1** (Standard Monomial Basis). *The term $\prod_{t\in D} X_t^{d_t}$ in Eq. (1) is a* monomial. *A polynomial $\Phi(\mathbf{X})$ is in standard monomial basis (SMB) when we keep only the terms with $c_{\mathbf{d}} \neq 0$ from Eq. (1).*

Unless othewise noted, we consider all polynomials to be in SMB representation. When it is unclear, we use $\mathrm{SMB}(\Phi)$ to denote the SMB form of a polynomial Φ.

Use def to define BIDB

spell it out

▶ **Definition 2.2** (Degree). *The degree of polynomial $\Phi(\mathbf{X})$ is the largest $\|\mathbf{d}\|_1$ such that $c_{(d_1,\ldots,d_n)} \neq 0$.*

Maybe use D' as set of base tuples for BIDB to distinguish from D for TIDB

As an example, the degree of the polynomial $X^2 + 2XY^2 + Y^2$ is 3. Product terms in lineage arise only from join operations (Fig. 1), so intuitively, the degree of a lineage polynomial is analogous to the largest number of joins needed to produce a result tuple. We call a polynomial $\Phi(\mathbf{X})$ a c-TIDB-*lineage polynomial* (or simply lineage polynomial), if there exists a $\mathcal{RA}^+$ query Q, c-TIDB $\mathcal{D}$, and result tuple t such that $\Phi(\mathbf{X}) = \Phi[Q, D, t](\mathbf{X})$.

2.1.1 c-TIDBs and 1-BIDBs

should be a subsec not sub subsec

An *incomplete database* Ω is a set of deterministic databases ω called possible worlds.

Why not use W (not $\{0,\ldots,c\}^D$)?

no need to redefine

A c-TIDB $\mathcal{D}$ is a pair $\left(\{0,\ldots,c\}^D, \mathcal{P}\right)$ such that $\{0,\ldots,c\}^D$ is an incomplete database whose set of possible worlds is the $c+1^n$ tuple/multiplicity combinations across all $t \in D$, where $|D| = n$, $D = \bigcup_{\mathbf{M}\in\{0,\ldots,c\}^D,\ \mathbf{M}_t \geq 1} t$ is the set of possible tuples across possible worlds, and $\mathcal{P}$ is a probability distribution over $\{0,\ldots,c\}^D$.

A block independent database (BIDB) is a related probabilistic data model $\mathcal{D} = (\Omega, \mathcal{P})$ such that the base set of tuples $D = \bigcup_{\omega\in\Omega,\ t\in\omega} t$ is partitioned into a set of n independent

Define BIDB "directly" as we did for c-TIDB i.e. of the form $(\{..\}^{D'}, P')$ allow P' to assign certain worlds 0 prob.

→ Current defn for 1-BIDB seems to indicate {0,1} only which is NOT enough for the redux

~~no need to define this since we do not need to make it clear that we've allowed for mult. here~~

blocks $\left\{(b_t)_{t\in[n]}\right\}$ such that the set of tuples $\left\{(t_j)_{j\in[|b|]}\right\}$ in block b_t are disjoint from one another. This construction produces the set of possible worlds Ω that consists of all unique combinations of tuples in D with the constraint that for any $\omega\in\Omega$, no two tuples $t_j, t_{j'}, j\neq j'$ from the same block b_t exist together. A c-BIDB has the further requirement that each block has a multiplicity of at most c. We present a reduction that is useful in producing our results:

Why does BIDB def have variables in it?

confusing.

▶ **Definition 2.3** (c-TIDB reduction). *Given c-TIDB $\mathcal{D} = \left(\{0,\ldots,c\}^D, \mathcal{P}\right)$, let $\mathcal{D}' = (\Omega, \mathcal{P}')$ be the 1-BIDB obtained in the following manner: for each $t\in D$, create block $b_t = \left\{\langle t, jX_{t,j}\rangle_{j\in[c]}\right\}$, such that $X_{t,j}\in\{0,1\}$. The probability distribution $\mathcal{P}'$ is the one induced by $\mathbf{p} = \left((p_{t,j})_{t\in D, j\in[c]}\right)$ and the BIDB disjoint requirement.*

Use $\mathcal{D}'$ for BIDB

~~need to explicitly spell this out~~

(1) Restate the full def of Φ (we have space for it)

For the c-TIDB $\mathcal{D}$, each $X_t\in[c]$, while in the reduced 1-BIDB $\mathcal{D}'$, each $X_{t,j}\in\{0,1\}$. Hence, in the setting of 1-BIDB, the base case of Fig. 1 now becomes $\Phi[R,D,t] = \sum_{j\in[c]} jX_{t,j}$. Then given the disjoint requirement and the semantics for constructing the lineage polynomial over a 1-BIDB, $\Phi[R,D',t]$ is of the same structure as the reformulated polynomial Φ_R of step i) from Definition 1.3, which then implies that $\widetilde{\Phi}$ is the reduced polynomial that results from step ii) of Definition 1.3, and further that Lemma 1.4 immediately follows for 1-BIDB polynomials: $\mathbb{E}_{\mathbf{W}\sim\mathcal{P}'}\left[\Phi\left(\mathbf{W}\right)\right] = \widetilde{\Phi}\left(\mathbf{p}\right)$.

Using Φ for both TIDB & BIDB

Aaron says: @atri, not sure if $\mathcal{P}'$ should be $\mathcal{P}''$ (in the above expectation) as discussed below. Since $\mathcal{P}' \equiv \mathcal{P}''$, then the proof still holds for Lemma 1.4, but maybe it is important to $\mathcal{P}''$ to drive the point home that we iterate over the all worlds set (as opposed to the set of possible worlds) when computing the expectation of a polynomial. Or maybe it suffices to note that $\mathcal{P}' \equiv \mathcal{P}''$.

(2) This redux is informal. Define in terms of D' tuples ie base case should be $\Phi[R, D', t_j] = j\cdot X_{t,j}$

If you do not start from possible world semantics this is not an issue. You just start w/ P''

Instead of looking only at the possible worlds of $\mathcal{D}$, one can consider all worlds, including those that cannot exist due to disjointness. The all worlds set can be modeled by $\mathbf{M}\in\{0,1\}^{cn}$,[3] such that $\mathbf{M}_{t,j}\in\mathbf{M}$ represents whether or not the multiplicity of t is j.We denote a probability distribution over all $\mathbf{M}\in\{0,1\}^n$ as $\mathcal{P}''$. When $\mathcal{P}''$ is the one induced from each $p_{t,j}$ while assigning $Pr\left[\mathbf{M}\right]=0$ for any $\mathbf{M}$ with $\mathbf{M}_{t,j}=\mathbf{M}_{t,j'}=1$ for $j\neq j'$, we end up with a bijective mapping from $\mathcal{P}'$ to $\mathcal{P}''$, such that each mapping is equivalent, implying the distributions are equivalent. ~~Appendix B.2 has more details.~~

Let $|\Phi|$ be the number of operators in Φ.

not defined

▶ **Corollary 2.4.** *If Φ is a ~~BIDB-lineage polynomial~~ already in SMB, then the expectation of Φ, i.e., $\mathbb{E}\left[\Phi\right] = \widetilde{\Phi}\left(p_1,\ldots,p_n\right)$ can be computed in $O\left(|\Phi|\right)$ time.*

~~start a subsection on possible world semantics~~

In some sense this section is not super relevant to paper. So putting all this in a separate sub-sect section makes sense

~~Queries over probabilistic databases are evaluated using the so-called possible world semantics.~~ Under the possible world semantics, the result of a query Q over an incomplete database Ω is the set of query answers produced by evaluating Q over each possible world $\omega\in\Omega$: $\{Q\left(\omega\right):\omega\in\Omega\}$.

This is where you can make the connection b/w our defn & possible world semantics

subscript

free

The result of a query is the pair $\left(Q\left(\omega\right),\mathcal{P}'\right)$ where $\mathcal{P}'$ is a probability distribution that assigns to each possible query result the sum of the probabilites of the worlds that produce this answer: $Pr\left[\omega\in\Omega\right] = \sum \omega'\in\Omega, Q\left(\omega'\right) = Q\left(\omega\right) Pr\left[\omega'\right]$.

Recalling Fig. 1 again, which defines the lineage polynomial $\Phi[Q,D,t]$ for any $\mathcal{RA}^+$ query. We now make a meaningful connection between possible world semantics and world assignments on the lineage polynomial.

[3] Here and later, especially in Sec. 4, we will rename the variables as $X_1,\ldots,X_n$, where $n=\sum_{i=1}^{\ell}|b_i|$.

~~This should be in main text & highlighted~~

~~Note: Don't forget to change the opening of S1 to not use the term product distribution, but rather state that $ \mathcal{P}$ is a probability distribution.~~

w/P

not defined

▶ **Proposition 2.5** (Expectation of polynomials). *Given a bag-PDB $\mathcal{D} = (\Omega, \mathcal{P})$, $\mathcal{RA}^+$ query Q, and lineage polynomial $\Phi[Q, D, t]$ for arbitrary result tuple t, we have (denoting $\mathbf{D}$ as the random variable over Ω):* $\mathbb{E}_{\mathbf{D}\sim\mathcal{P}}[Q(\mathbf{D})(t)] = \mathbb{E}_{\mathbf{W}\sim\mathcal{P}}\left[\Phi[Q, D, t]\left(\mathbf{W}\right)\right]$.

move to next section?

A formal proof of Proposition 2.5 is given in Appendix B.3.[4] We focus on the problem of computing $\mathbb{E}_{\mathcal{P}}\left[\Phi[Q, D, t]\left(\mathbf{W}\right)\right]$ from now on, assume implicit Q, D, t, and drop them from $\Phi[Q, D, t]$ (i.e., $\Phi\left(\mathbf{X}\right)$ will denote a polynomial).

2.2 Formalizing Problem 1.6

Need a filler sentence remind reader about P1.6

make sure intro has fw ptr here

We represent lineage polynomials via *arithmetic circuits* [9], a standard way to represent polynomials over fields (particularly in the field of algebraic complexity) that we use for polynomials over $\mathbb{N}$ in the obvious way. Since we are particularly using circuits to model lineage polynomials, we can refer to these circuits as lineage circuits. However, when the meaning is clear, we will drop the term lineage and only refer to them as circuits.

$x = (X_1, \ldots, X_n)$

▶ **Definition 2.6** (Circuit). *A circuit $\mathtt{C}$ is a Directed Acyclic Graph (DAG) whose source gates (in degree of 0) consist of elements in either $\mathbb{N}$ or $\mathbf{X}$. For each result tuple there exists one sink gate. The internal gates have binary input and are either sum (+) or product (×) gates. Each gate has the following members: $\mathtt{type}$, $\mathtt{partial}$, $\mathtt{input}$, $\mathtt{degree}$, $\mathtt{Lweight}$, and $\mathtt{Rweight}$, where $\mathtt{type}$ is the value type $\{+, \times, \textsc{var}, \textsc{num}\}$ and $\mathtt{input}$ the list of inputs. Source gates have an extra member $\mathtt{val}$ storing the value. $\mathtt{C}_\mathtt{L}$ ($\mathtt{C}_\mathtt{R}$) denotes the left (right) input of $\mathtt{C}$.*

Aaron says: Does the following matter, i.e., does it point anything out special for our research?

When the underlying DAG is a tree (with edges pointing towards the root), the structure is an expression tree $\mathtt{T}$. In such a case, the root of $\mathtt{T}$ is analogous to the sink of $\mathtt{C}$. The fields $\mathtt{partial}$, $\mathtt{degree}$, $\mathtt{Lweight}$, and $\mathtt{Rweight}$ are used in the proofs of Appendix D.

The circuits in Fig. 2 encode their respective polynomials in column Φ. Note that each circuit $\mathtt{C}$ encodes a tree, with edges pointing towards the root.

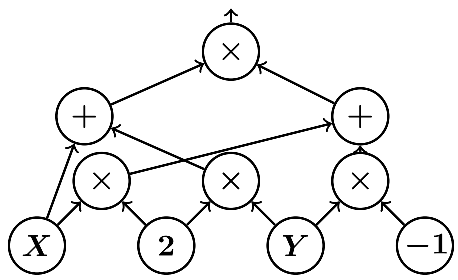

Figure 3 Circuit encoding of $(X + 2Y)(2X - Y)$

We next formally define the relationship of circuits with polynomials. While the definition assumes one sink for notational convenience, it easily generalizes to the multiple sinks case.

▶ **Definition 2.7** ($\textsc{poly}(\cdot)$). *Denote $\textsc{poly}(\mathtt{C})$ to be the function from the sink of circuit $\mathtt{C}$ to its corresponding polynomial (in SMB). $\textsc{poly}(\cdot)$ is recursively defined on $\mathtt{C}$ as follows, with addition and multiplication following the standard interpretation for polynomials:*

$$\textsc{poly}(\mathtt{C}) = \begin{cases} \textsc{poly}(\mathtt{C}_\mathtt{L}) + \textsc{poly}(\mathtt{C}_\mathtt{R}) & \text{if } \mathtt{C}.\mathtt{type} = + \\ \textsc{poly}(\mathtt{C}_\mathtt{L}) \cdot \textsc{poly}(\mathtt{C}_\mathtt{R}) & \text{if } \mathtt{C}.\mathtt{type} = \times \\ \mathtt{C}.\mathtt{val} & \text{if } \mathtt{C}.\mathtt{type} = \textsc{var} \text{ OR } \textsc{num}. \end{cases}$$

[4] Although Proposition 2.5 follows, e.g., as an obvious consequence of [28]'s Theorem 7.1, we are unaware of any formal proof for bag-probabilistic databases.

C need not encode $\Phi(\mathbf{X})$ in the same, default SMB representation. For instance, C could encode the factorized representation $(X + 2Y)(2X - Y)$ of $\Phi(\mathbf{X}) = 2X^2 + 3XY - 2Y^2$, as shown in Fig. 3, while $\text{POLY}(\texttt{C}) = \Phi(\mathbf{X})$ is always the equivalent SMB representation.

▶ **Definition 2.8** (Circuit Set). *$\texttt{CSet}(\Phi(\mathbf{X}))$ is the set of all possible circuits C such that $\text{POLY}(C) = \Phi(\mathbf{X})$.*

The circuit of Fig. 3 is an element of $\texttt{CSet}\left(2X^2 + 3XY - 2Y^2\right)$. One can think of $\texttt{CSet}(\Phi(\mathbf{X}))$ as the infinite set of circuits where for each element C, $\text{POLY}(\texttt{C}) = \Phi(\mathbf{X})$.

We are now ready to formally state the final version of Problem 1.6.

▶ **Definition 2.9** (The Expected Result Multiplicity Problem). *Let $\mathcal{D}$ be an arbitrary* BIDB-*PDB and* $\mathbf{X}$ *be the set of variables annotating tuples in D_Ω. Fix an $\mathcal{RA}^+$ query Q and a result tuple t. The* Expected Result Multiplicity Problem *is defined as follows:*

Input: *$C \in \texttt{CSet}(\Phi(\mathbf{X}))$ for $\Phi(\mathbf{X}) = \Phi[Q, D, t]$* ***Output:*** *$\mathbb{E}_{\mathbf{W}\sim\mathcal{P}}[\Phi[Q, D, t](\mathbf{W})]$*

~~make sure intro has a flow to there~~

2.3 Relationship to Deterministic Query Runtimes

Filler sentence to connect back to intro

To decouple our results from specific join algorithms, we first abstract the cost of a join.

▶ **Definition 2.10** (Join Cost). *Denote by $T_{join}(R_1, \ldots, R_m)$ the runtime of an algorithm for computing the m-ary join $R_1 \bowtie \ldots \bowtie R_m$. We require only that the algorithm must enumerate its output, i.e., that $T_{join}(R_1, \ldots, R_m) \geq |R_1 \bowtie \ldots \bowtie R_m|$.*

Use D instead of D

Worst-case optimal join algorithms [37, 36] and query evaluation via factorized databases [39] (as well as work on FAQs [33]) can be modeled as $\mathcal{RA}^+$ queries (though the query size is data dependent). For these algorithms, $T_{join}(R_1, \ldots, R_n)$ is linear in the *AGM bound* [6]. Our cost model for general query evaluation follows from the join cost:

$$T_{det}(R, D) = |D.R| \quad T_{det}(\sigma Q, D) = T_{det}(Q, D) \quad T_{det}(\pi Q, D) = T_{det}(Q, D) + |Q(D)|$$
$$T_{det}(Q \cup Q', D) = T_{det}(Q, D) + T_{det}(Q', D) + |Q(D)| + |Q'(D)|$$
$$T_{det}(Q_1 \bowtie \ldots \bowtie Q_m, D) = T_{det}(Q_1, D) + \ldots + T_{det}(Q_m, D) + T_{join}(Q_1(D), \ldots, Q_m(D))$$

Under this model, an $\mathcal{RA}^+$ query Q evaluated over database D has runtime $O(T_{det}(Q, D))$. We assume that full table scans are used for every base relation access. We can model index scans by treating an index scan query $\sigma_\theta(R)$ as a base relation.

Finally, Lemma E.2 and Lemma E.3 show that for any $\mathcal{RA}^+$ query Q and D_Ω, there exists a circuit $\texttt{C}^*$ such that $T_{LC}(Q, D_\Omega, C^*)$ and $|\texttt{C}^*|$ are both $O(T_{det}(Q, D_\Omega))$. Recall we assumed these two bounds when we moved from Problem 1.5 to Problem 1.6.

correct notation.

3 Hardness of Exact Computation

In this section, we will prove the hardness results claimed in Table 1 for a specific (family) of hard instance $(Q, \mathcal{D})$ for Problem 1.2 where $\mathcal{D}$ is a 1-TIDB. Note that this implies hardness for c-TIDBs ($c \geq 1$), BIDBs and general bag-PDB, showing Problem 1.2 cannot be done in $O(T^*_{det}(Q, D))$ runtime.

3.1 Preliminaries

Our hardness results are based on (exactly) counting the number of (not necessarily induced) subgraphs in G isomorphic to H. Let $\#(G, H)$ denote this quantity. We can think of H