Ok, at this point it is probably easiest for me to draft this.

Aaron says: I am *unsure* of this footnote. @atri may need to word smith this one. I don't feel like I entirely understand the purpose of this footnote. E.g., we could have a query that runs deterministically in $\Omega_k(n)$ worst case time; but this doesn't mean that $T^*(Q, \mathcal{D})$ doesn't have a worst case lower bound of $\Omega(n)^{c_0}$, correct? We would replace $T_{det}(Q, D, c)$ with $\Omega_k(n)$, no? If we replace $T_{det}(Q, D, c)$ with n, then this doesn't accurately reflect the worst case lower bound for counting k-cliques in the first place.

I don't see why you say this.

already imply the claimed lower bounds if we were to replace the $T_{det}(\text{OPT}(Q), D, c)$ by just n (indeed these results follow from known lower bounds for deterministic query processing). Our contribution is to then identify a family of hard queries where deterministic query processing is 'easy' but computing the expected multiplicities is hard.

Our upper bound results. We introduce a $(1 \pm \epsilon)$-approximation algorithm that computes Problem 1.1 in time $O_\epsilon(T_{det}(\text{OPT}(Q), D, c))$. This means, when we are okay with approximation, that we solve Problem 1.1 in time linear in the size of the deterministic query and bag PDBs are deployable in practice. In contrast, known approximation techniques ([40, 32]) in set-PDBs need time $\Omega(T_{det}(\text{OPT}(Q), D, c)^{2k})$ (see Appendix G). Further, our approximation algorithm works for a more general notion of bag PDBs beyond c-TIDBs (see Sec. 2.2).

1.1 Polynomial Equivalence

A common encoding of probabilistic databases (e.g., in [30, 29, 5, 2] and many others) relies on annotating tuples with lineages or propositional formulas that describe the set of possible worlds that the tuple appears in. The bag semantics analog is a provenance/lineage polynomial (see Fig. 1) $\Phi[Q, D, t]$ [27], a polynomial with non-zero integer coefficients and exponents, over integer variables $\mathbf{X}$ encoding input tuple multiplicities.

Aaron says: This seems confusing since I thought the goal was to have $\mathbf{X}$ be abstract/typeless.

X should be typeless. You can say when we evaluate this poly X get replaced by int. values.

We drop Q, D, and t from $\Phi[Q, D, t]$ when they are clear from the context or irrelevant to the discussion. We now specify the problem of computing the expectation of tuple multiplicity in the language of lineage polynomials:

database D that counts the number of k-cliques, the results show a deterministic runtime of $\Omega_k(n)$, implying our lower bounds would hold.

$$\Phi[\pi_A(Q), \overline{D}, t] = \sum_{t' : \pi_A(t') = t} \Phi[Q, \overline{D}, t'] \qquad \Phi[Q_1 \cup Q_2, \overline{D}, t] = \Phi[Q_1, \overline{D}, t] + \Phi[Q_2, \overline{D}, t]$$

$$\Phi[\sigma_\theta(Q), \overline{D}, t] = \begin{cases} \Phi[Q, \overline{D}, t] & \text{if } \theta(t) \\ 0 & \text{otherwise.} \end{cases} \qquad \begin{aligned} \Phi[Q_1 \bowtie Q_2, \overline{D}, t] =& \Phi[Q_1, \overline{D}, \pi_{attr(Q_1)} t] \\ & \cdot \Phi[Q_2, \overline{D}, \pi_{attr(Q_2)} t] \\ \Phi[R, \overline{D}, t] =& X_t \end{aligned}$$

Figure 1 Construction of the lineage (polynomial) for an $\mathcal{RA}^+$ query Q over an arbitrary deterministic database $\overline{D}$, where $\mathbf{X}$ consists of all X_t over all R in $\overline{D}$ and t in R. Here $\overline{D}.R$ denotes the instance of relation R in $\overline{D}$. Please note, after we introduce the reduction to 1-BIDB, the base case will be expressed alternatively.

▶ **Problem 1.2** (Expected Multiplicity of Lineage Polynomials). *Given an $\mathcal{RA}^+$ query Q, c-TIDB $\mathcal{D}$ and result tuple t, compute the expected multiplicity of the polynomial $\Phi[Q,D,t]$ (i.e., $\mathbb{E}_{\mathbf{W}\sim\mathcal{P}}\left[\Phi[Q,D,t](\mathbf{W})\right]$, where $\mathbf{W}\in\{0,\ldots,c\}^D$).*

We note that computing Problem 1.1 is equivalent (yields the same result as) to computing Problem 1.2 (see Proposition 2.8).

All of our results rely on working with a *reduced* form $\left(\widetilde{\Phi}\right)$ of the lineage polynomial Φ. In fact, it turns out that for the 1-TIDB case, computing the expected multiplicity (over bag query semantics) is *exactly* the same as evaluating this reduced polynomial over the probabilities that define the 1-TIDB. This is also true when the query input(s) is a block independent disjoint probabilistic database [40] (bag query semantics with tuple multiplicity at most 1), for which the proof of Lemma 1.4 (introduced shortly) holds . Next, we motivate this reduced polynomial. Consider the query Q_1 defined as follows over the bag relations of Fig. 2:

```
SELECT DISTINCT 1 FROM T t₁, R r, T t₂
WHERE t₁.Point = r.Point₁ AND t₂.Point = r.Point₂
```

It can be verified that $\Phi\left(A,B,C,E,X,Y,Z\right)$ for the sole result tuple of Q_1 is $AXB+BYE+BZC$. Now consider the product query $Q_1^2 = Q_1\times Q_1$. The lineage polynomial for Q_1^2 is given by $\Phi_1^2\left(A,B,C,E,X,Y,Z\right)$

$$=A^2X^2B^2+B^2Y^2E^2+B^2Z^2C^2+2AXB^2YE+2AXB^2ZC+2B^2YEZC.$$

To compute $\mathbb{E}\left[\Phi_1^2\right]$ we can use linearity of expectation and push the expectation through each summand. To keep things simple, let us focus on the monomial $\Phi_1^{(ABX)^2}=A^2X^2B^2$ as the procedure is the same for all other monomials of Φ_1^2. Let W_X be the random variable corresponding to a lineage variable X. Because the distinct variables in the product are independent, we can push expectation through them yielding $\mathbb{E}\left[W_A^2W_X^2W_B^2\right]=\mathbb{E}\left[W_A^2\right]\mathbb{E}\left[W_X^2\right]\mathbb{E}\left[W_B^2\right]$. Since $W_A,W_B\in\{0,1\}$ we can further derive $\mathbb{E}\left[W_A\right]\mathbb{E}\left[W_X^2\right]\mathbb{E}\left[W_B\right]$ by the fact that for any $W\in\{0,1\}$, $W^2=W$. Observe that if $X\in\{0,1\}$, then we further would have $\mathbb{E}\left[W_A\right]\mathbb{E}\left[W_X\right]\mathbb{E}\left[W_B\right]=p_A\cdot p_X\cdot p_B$ (denoting $Pr\left[W_A=1\right]=p_A$) $=\widetilde{\Phi}_1^{(ABX)^2}\left(p_A,p_X,p_B\right)$ (see *ii*) of Definition 1.3). However, in this example, we get stuck with $\mathbb{E}\left[W_X^2\right]$, since $W_X\in\{0,1,2\}$ and for $W_X\leftarrow 2$, $W_X^2\neq W_X$.

Denote the variables of Φ to be $\textsc{Vars}\left(\Phi\right)$. In the c-TIDB setting, $\Phi\left(\mathbf{X}\right)$ has an equivalent reformulation $\left(\Phi_R\left(\mathbf{X_R}\right)\right)$ that is of use to us, where $|\mathbf{X_R}|=c\cdot|\mathbf{X}|$. Given $X_t\in\textsc{Vars}\left(\Phi\right)$, by definition $X_t\in\{0,\ldots,c\}$. We can replace X_t by $\sum_{j\in[c]}jX_{t,j}$ where the variables $\left(X_{t,j}\right)_{j\in[c]}$ are disjoint and each $X_{t,j}\in\{0,1\}$. Then for any $\mathbf{W}\in\{0,\ldots,c\}^D$ and corresponding reformulated world $\mathbf{W_R}\in\{0,1\}^{Dc}$, we set $\mathbf{W}_{\mathbf{R}_{t,j}}=1$ for $\mathbf{W}_t=j$, while $\mathbf{W}_{\mathbf{R}_{t,j'}}=0$ for all $j'\neq j\in[c]$. By construction then $\Phi\left(\mathbf{X}\right)\equiv\Phi_R\left(\mathbf{X_R}\right)$ $\left(\mathbf{X_R}=\textsc{Vars}\left(\Phi_R\right)\right)$ since for any valuation $X_t\in[c]$ we have the equality $X_t=j=\sum_{j\in[c]}jX_j$.

Aaron says: I don't know the rules here, but since we have already (informally) defined $\mathbf{X}$ to be variables of type integer encoding multiplicities (see todo note above) and thus worlds, it seems that it is fine and natural to refer to valuations of the variables themselves, without having to use $\mathbf{W}$ necessarily. The point I am trying to get across in the last sentence is, given these semantics and domains, we have an equivalent polynomial. Or is it wrong to use $\mathbf{X}$ and we should rather say, "for any $\mathbf{W}\in\{0,\ldots,c\}^D$, $\mathbf{W_R}\in\{0,1\}^{Dc}$ we have that $\mathbf{W}_t=j=\sum_{j\in[c]}j\cdot\mathbf{W}_{\mathbf{R}_{t,j}}$?

Yeah the above discussion can be done just in terms of X

Considering again our example,

$$\Phi_{1,R}^{(ABX)^2}(A,X,B) = \Phi_1^{(AXB)^2}\left(\sum_{j_1\in[c]} j_1A_{j_1}, \sum_{j_2\in[c]} j_2X_{j_2}, \sum_{j_3\in[c]} j_3B_{j_3}\right)$$

$$= \left(\sum_{j_1\in[c]} j_1A_{j_1}\right)^2 \left(\sum_{j_2\in[c]} j_2X_{j_2}\right)^2 \left(\sum_{j_3\in[c]} j_3B_{j_3}\right)^2.$$

Since the set of multiplicities for tuple t by nature are disjoint we can drop all cross terms and have $\Phi^2_{1,R} = \sum_{j_1,j_2,j_3\in[c]} j_1^2A_{j_1}^2 j_2^2X_{j_2}^2 j_3^2B_{j_3}^2$. Computing expectation we get $\mathbb{E}\left[\Phi^2_{1,R}\right] = \sum_{j_1,j_2,j_3\in[c]} j_1^2j_2^2j_3^2\, \mathbb{E}\left[W_{A_{j_1}}\right]\mathbb{E}\left[W_{X_{j_2}}\right]\mathbb{E}\left[W_{B_{j_3}}\right]$, since we now have that all $W_{X_j}\in\{0,1\}$. This leads us to consider a structure related to the lineage polynomial.

▶ **Definition 1.3.** *For any polynomial* $\Phi\left((X_t)_{t\in D}\right)$ *define the reformulated polynomial* $\Phi_R\left((X_{t,j})_{t\in D, j\in[c]}\right)$ *to be the polynomial* $\Phi_R = \Phi\left(\left(\sum_{j\in[c]} j\cdot X_{t,j}\right)_{t\in D}\right)$ *and ii) define the reduced polynomial* $\widetilde{\Phi}\left((X_{t,j})_{t\in D,j\in[c]}\right)$ *to be the polynomial resulting from converting* Φ_R *into the standard monomial basis (SMB),* [4] *removing all monomials containing the term* $X_{t,j}X_{t,j'}$ *for* $t\in D, j\neq j'\in[c]$, *and setting all* variable *exponents* $e>1$ *to* 1.

Continuing with the example [5] $\Phi_1^2(A,B,C,E,X_1,X_2,Y,Z)$ we have

$$\widetilde{\Phi}_1^2(A,B,C,E,X_1,X_2,Y,Z) =$$

$$A\left(\sum_{j\in[c]} j^2X_j\right)B + BYE + BZC + 2A\left(\sum_{j\in[c]} j^2X_j\right)BYE + 2A\left(\sum_{j\in[c]} j^2X_j\right)BZC + 2BYEZC =$$

$$ABX_1 + AB(2)^2X_2 + BYE + BZC + 2AX_1BYE + 2A(2)^2X_2BYE + 2AX_1BZC + 2A(2)^2X_2BZC + 2BYEZC.$$

Note that we have argued that for our specific example the expectation that we want is $\widetilde{\Phi}_1^2(Pr(A=1), Pr(B=1), Pr(C=1)), Pr(E=1), Pr(X_1=1), Pr(X_2=1), Pr(Y=1), Pr(Z=1))$. Lemma 1.4 generalizes the equivalence to *all* $\mathcal{RA}^+$ queries on c-TIDBs (proof in Appendix B.5).

▶ **Lemma 1.4.** *For any* c-TIDB $\mathcal{D}$, $\mathcal{RA}^+$ *query* Q, *and lineage polynomial* $\Phi(\mathbf{X}) = \Phi[Q,D,t](\mathbf{X})$, *it holds that* $\mathbb{E}_{\mathbf{W}\sim\mathcal{P}}\left[\Phi_R(\mathbf{W})\right] = \widetilde{\Phi}(\mathbf{p})$, *where* $\mathbf{p} = \left((p_{t,j})_{t\in D,j\in[c]}\right)$.

1.2 Our Techniques

Lower Bound Proof Techniques.

Aaron says: Regarding what follows (in the next paragraph): I think this *may* be misleading (also, technically incorrect since Φ is used instead of $\widetilde{\Phi}$) since it the *lead* c_{2k} of the term in $\widetilde{\Phi}(\mathbf{X})$ with $2k$ distinct variables. However, technically, since we have that $\widetilde{\Phi}(\mathbf{p})$ is a univariate polynomial, then, indeed this IS an accurate statement, since the term with $2k$ distinct variables in $\widetilde{\Phi}(\mathbf{p})$ is the term with the highest degree (this assumes for d distinct edges that $d\geq k$ for our special graph query; otherwise, there is no k-matching, and the leading coefficient is not c_{2k}). Perhaps we should note this. However, the context is in light of considering the *univariate* polynomial $\widetilde{\Phi}(\mathbf{p})$. Perhaps change Φ to $\widetilde{\Phi}(p,\ldots,p)$.

already make sense. Looks like you did this?

[4] This is the representation, typically used in set-PDBs, where the polynomial is reresented as sum of 'pure' products. See Definition 2.1 for a formal definition.

[5] To save clutter we do not show the full expansion for variables with greatest multiplicity $=1$ since e.g. for variable A, the sum of products itself evaluates to $1^2\cdot A^2 = A$.

However, systems can directly emit compact, factorized representations of $\Phi(\mathbf{X})$ (e.g., as a consequence of the standard projection push-down optimization [25]). For example, in Fig. 2, $B(Y+Z)$ is a factorized representation of the SMB-form $BY + BZ$. Accordingly, this work uses (arithmetic) circuits[6] as the representation system of $\Phi(\mathbf{X})$.

Given that there exists a representation $\mathtt{C}^*$ such that $T_{LC}(Q, D, \mathtt{C}^*) \leq O\left(T_{det}\left(\text{OPT}\left(Q\right), D, c\right)\right)$, we can now focus on the complexity of the EC step. We can represent the factorized lineage polynomial by its correspoding arithmetic circuit $\mathtt{C}$ (whose size we denote by $|\mathtt{C}|$). As we also show in Appendix E.2.2, this size is also bounded by $T_{det}\left(\text{OPT}\left(Q\right), D, c\right)$ (i.e., $|\mathtt{C}^*| \leq O\left(T_{det}\left(\text{OPT}\left(Q\right), D, c\right)\right)$). Thus, the question of approximation can be stated as the following stronger (since Problem 1.5 has access to *all* equivalent $\mathtt{C}$ representing $Q\left(\mathbf{W}\right)(t)$), but sufficient condition:

▶ **Problem 1.6.** *Given one circuit $\mathtt{C}$ that encodes $\Phi[Q, D, t]$ for all result tuples t (one sink per t) for c-TIDB $\mathcal{D}$ and $\mathcal{RA}^+$ query Q, does there exist an algorithm that computes a $(1 \pm \epsilon)$-approximation of $\mathbb{E}_{\mathbf{W}\sim\mathcal{P}}\left[Q\left(\mathbf{W}\right)(t)\right]$ (for all result tuples t) in $O\left(|\mathtt{C}|\right)$ time?*

For an upper bound on approximating the expected count, it is easy to check that if all the probabilties are constant then $\Phi\left(p_1, \ldots, p_n\right)$ (i.e. evaluating the original lineage polynomial over the probability values) is a constant factor approximation. For example, using Q_1^2 from above (with $c = 1$) and p_A to denote $Pr\left[A = 1\right]$, we can see that

Sounds good

Aaron says: I changed Problem 1.6 to use c-TIDB. Correct me if this is wrong. Our results do apply to a more general class of bag-PDB, but the main data model considered in this paper is c-TIDB.
Also, for the example above and worked out in what follows, it might be better flow to keep $c = 2$ and change what is below.

$$\begin{aligned}\Phi_1^2\left(\mathbf{p}\right) &= p_A^2 p_X^2 p_B^2 + p_B^2 p_Y^2 p_E^2 + p_B^2 p_Z^2 p_C^2 + 2p_A p_X p_B^2 p_Y p_E + 2p_A p_X p_B^2 p_Z p_C + 2p_B^2 p_Y p_E p_Z p_C \\ &\leq p_A p_X p_B + p_B p_Y p_E + p_B p_Z p_C + 2p_A p_X p_B p_Y p_E + 2p_A p_X p_B p_Z p_C + 2p_B p_Y p_E p_Z p_C = \widetilde{\Phi}_1^2\left(\mathbf{p}\right)\end{aligned}$$

If we assume that all seven probability values are at least $p_0 > 0$, we get that $\Phi_1^2\left(\mathbf{p}\right)$ is in the range $\left[\left(p_0\right)^3 \cdot \widetilde{\Phi}_1^2\left(\mathbf{p}\right), \widetilde{\Phi}_1^2\left(\mathbf{p}\right)\right]$, which is *not a tight approximation.* In sec. 4 we demonstrate that a $(1 \pm \epsilon)$ (multiplicative) approximation with competitive performance is achievable. To get an $(1 \pm \epsilon)$-multiplicative approximation and solve Problem 1.6, using $\mathtt{C}$ we uniformly sample monomials from the equivalent SMB representation of Φ (without materializing the SMB representation) and 'adjust' their contribution to $\widetilde{\Phi}\left(\cdot\right)$.

Applications. Recent work in heuristic data cleaning [51, 45, 42, 8, 45] 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 [37], they are impractically slow [19, 18], 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.

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.

[6] An arithmetic circuit is a DAG with variable and/or numeric source nodes and internal, each nodes representing either an addition or multiplication operator.

This setup is not quite right. First of all Prop 2.4 has nothing to do with a circuit. The statement should go as follows:

(i) Let C be a ckt for a c-TIDB

(ii) Then ∃ an equivalent ckt C' for Binary-BIDB s.t. $\gamma(C') \leq 1-(c+1)^{-(k-1)}$

(iii) state how the depth & size changes

Given the above, the algorithm is a sampling based algorithm for the above sum: we sample (via SampleMonomial) $(\mathtt{v}, \mathtt{c}) \in \mathtt{E}(\mathtt{C})$ with probability proportional to $|\mathtt{c}|$ and compute $\mathtt{Y} = \mathbb{1}_{\text{ISIND}(\mathtt{v}_\mathtt{m})} \cdot \prod_{X_i \in \mathtt{v}} p_i$. Repeating the sampling an appropriate number of times and computing the average of $\mathtt{Y}$ gives us our final estimate. OnePass is used to compute the sampling probabilities needed in SampleMonomial (details are in Appendix D).

Runtime analysis. We can argue the following runtime for the algorithm outlined above:

Only in the proof do you need to bring up Prop 2.4.

▶ **Theorem 4.7.** *Let $\mathcal{C}$ be an arbitrary* Binary-BIDB *circuit, define $\Phi(\mathbf{X}) = \textsc{poly}(\mathcal{C})$, let $k = \textsc{deg}(\mathcal{C})$, and let $\gamma = \gamma(\mathcal{C})$. Further let it be the case that $p_i \geq p_0$ for all $i \in [n]$. Then an estimate $\mathcal{E}$ of $\widetilde{\Phi}(p_1, \ldots, p_n)$ satisfying*

$$Pr\left(\left|\mathcal{E} - \widetilde{\Phi}(p_1, \ldots, p_n)\right| > \epsilon' \cdot \widetilde{\Phi}(p_1, \ldots, p_n)\right) \leq \delta \tag{3}$$

Specifically state how C' is built for C i.e. replace the input gate X_t by the ckt for $\sum_{j\geq 1} X_{j,t}$

I'm assuming the pf will go to the appendix?

can be computed in time

$$O\left(\left(\textsc{size}(\mathcal{C}) + \frac{\log\frac{1}{\delta} \cdot k \cdot \log k \cdot \textsc{depth}(\mathcal{C}))}{(\epsilon')^2 \cdot (1-\gamma)^2 \cdot p_0^{2k}}\right) \cdot \overline{\mathcal{M}}\left(\log\left(|\mathcal{C}|(1, \ldots, 1)\right), \log\left(\textsc{size}(\mathcal{C})\right)\right)\right). \tag{4}$$

In particular, if $p_0 > 0$ and $\gamma < 1$ are absolute constants then the above runtime simplifies to $O_k\left(\left(\frac{1}{(\epsilon')^2} \cdot \textsc{size}(\mathcal{C}) \cdot \log\frac{1}{\delta}\right) \cdot \overline{\mathcal{M}}\left(\log\left(|\mathcal{C}|(1, \ldots, 1)\right), \log\left(\textsc{size}(\mathcal{C})\right)\right)\right)$.

The restriction on γ is satisfied by any 1-TIDB (where $\gamma = 0$ in the equivalent 1-BIDB of Proposition 2.4) as well as for all three queries of the PDBench BIDB benchmark (see Appendix D.10 for experimental results). Further, we can alo argue the following result:

▶ **Lemma 4.8.** *Given* Binary-BIDB *computed from the reduction of Proposition 2.4, $\gamma(\mathcal{C}) \leq 1-(c+1)^{-(k-1)}$.*

Argument just in terms of C & C'.

Needs to be re-written as above. Structure as follows:

Proof of Lemma 4.8. Let $\mathcal{D}' = \left(\bigtimes_{t \in D'} \{0, c_t\}, \mathcal{D}'\right)$ be the reduced Binary-BIDB and $\mathcal{D} = \left(\{0, \ldots, c\}^D, \mathcal{D}\right)$ the original c-TIDB.

By Proposition 2.4, $\mathcal{D}'$ is a Binary-BIDB. By Definition 2.3, a block B_t of $\mathcal{D}'$ has the property that $\sum_{t \in D, j \in [c]} p_{t,j} \leq 1$. Then, if we consider the case of strict inequality, we have an extra possible outcome in block B_t, the outcome when no tuple is present in a possible world. Let us denote this as t_0. Then there are at most $c+1$ disjoint tuples in B_t. We argue later that the case when t_0 is a possibility produces the worst case γ.

Let $\Phi'(\mathbf{X})$ be an aribitrary polynomial produced by $Q(\mathcal{D}')$ with $\mathbf{X} = (X_{t,j})_{t \in D', j \in [0,c]}$ the set of variables in $\mathcal{D}'$. Let m be an arbitrary monomial in $\Phi'(\mathbf{X})$ and v_m be the set of variables appearing in m. We define a cross term to be any monomial m such that there exists $j \neq j' \in [0, c]$ such that $X_{t,j}, X_{t,j'} \in v_m$.

Do not bring in Q here. Argue in terms of C & C'.

(1) Show the conversion for C to C' as above

The semantics of Fig. 3 show that a new monomial product can only be generated by the $\bowtie$ operator of $\mathcal{RA}^+$ queries. Further, a cross term may only be produced specifically when the join is a self join. The highest number of terms that can be produced by a self join of B_t is $(c+1)^k$, the case for when all tuples join and $\sum_{t \in D, j \in [c]} p_{t,c} < 1$ as noted above. For monomials $m \in \left\{\bigtimes_{i \in [k], j \in [0,c]} X_{t,j_i}\right\}$, there exist *exaclty* $(c+1)$ *non*-cross terms, specifically $X^k_{t,j}$ for $j \in [0, c]$. Then there are exactly $(c+1)^k - (c+1)$ cross terms (cancellations). This implies that $\gamma(\mathtt{C}) = 1 - \frac{(c+1)}{(c+1)^k}$ for this case.

(2) Then argue that

We now show that the case above is indeed the worst case. First, given a self join, it is always the case that $X^k_{t,j}$ will be in the output since all tuples join with themselves. Then, the most number of cancellations occurs when we have that all $X_{t,j}$ joins with all $X_{t,j'}$ for

the C → C' conversion follows from redux in Def 2.3 & hence by Prop 2.4, C' is a valid ckt wrt.

(state that the corresponding lineage polys have the same exp. mult) i.e. POLY(C) & POLY(C')

(3) Rest of the should be just in terms of C, C', POLY(C) & POLY(C').

Here's a better way to do the monomial argument:
(1) first consider $E(C)$ i.e. set of all expanded monomials for c-TIDB ckt C
(2) Fix arbit monomial m in C ⇒ then each monomial $d_i \dots d_\ell$ $X_{t_1}, \dots X_{t_\ell}$ gives us blob set of monomials in $E(C')$. Note that if we can show the cancellations among this batch is $\leq \gamma_0$, then same bound of γ_0 holds overall. agree

$j \neq j' \in [0, c]$. Finally, it is the case that $c^k - c \leq (c+1)^k - (c+1) = \sum_{i=1}^{k}\binom{k}{i}c^i - (c-1)$ for $c, k \in \mathbb{N}$, which implies that the worst case is when we have the 'extra' tuple t_0 and all tuples joining, which is exactly the case above, producing the greatest $\gamma(\mathtt{C})$ ratio.

Since the size of any block B is $c+1$, it follows that $\gamma(\mathtt{C})$ ratio for block B_t is the same when taken across all blocks of $Q(\mathcal{D}')$, since the number of blocks n cancels out of the ratio calculations.

(3) Observe that cancellations can only happen for monomials for each $d_i X_{t_i}$.

We briefly connect the runtime in Eq. (4) to the algorithm outline earlier (where we ignore the dependence on $\overline{\mathcal{M}}(\cdot,\cdot)$, which is needed to handle the cost of arithmetic operations over integers). The $\text{SIZE}(\mathtt{C})$ comes from the time take to run OnePass once (OnePass essentially computes $|\mathtt{C}|(1,\dots,1)$ using the natural circuit evaluation algorithm on $\mathtt{C}$). We make $\frac{\log\frac{1}{\delta}}{(\epsilon')^2\cdot(1-\gamma)^2\cdot p_0^{2k}}$ many calls to SampleMonomial (each of which essentially traces $O(k)$ random sink to source paths in $\mathtt{C}$ all of which by definition have length at most $\text{DEPTH}(\mathtt{C})$).

Finally, we address the $\overline{\mathcal{M}}(\log(|\mathtt{C}|(1,\dots,1)), \log(\text{SIZE}(\mathtt{C})))$ term in the runtime.

▶ **Lemma 4.9.** *For any Binary-BIDB circuit $\mathtt{C}$ with $\text{DEG}(\mathtt{C}) = k$, we have $|\mathtt{C}|(1,\dots,1) \leq 2^{2^k\cdot\text{DEPTH}(\mathtt{C})}$. Further, if $\mathtt{C}$ is a tree, then we have $|\mathtt{C}|(1,\dots,1) \leq \text{SIZE}(\mathtt{C})^{O(k)}$.*

Note that the above implies that with the assumption $p_0 > 0$ and $\gamma < 1$ are absolute constants from Theorem 4.7, then the runtime there simplifies to $O_k\left(\frac{1}{(\epsilon')^2}\cdot\text{SIZE}(\mathtt{C})^2\cdot\log\frac{1}{\delta}\right)$ for general circuits $\mathtt{C}$. If $\mathtt{C}$ is a tree, then the runtime simplifies to $O_k\left(\frac{1}{(\epsilon')^2}\cdot\text{SIZE}(\mathtt{C})\cdot\log\frac{1}{\delta}\right)$, which then answers Problem 1.6 with yes for such circuits.

Then do the argmt for $X_t^{d_t}$ as you do → γ is $\leq 1-(c+1)^{-(d_t-1)}$

Finally, note that by Proposition E.1 and Lemma E.2 for any $\mathcal{RA}^+$ query Q, there exists a circuit $\mathtt{C}^*$ for $\Phi[Q, D, t]$ such that $\text{DEPTH}(\mathtt{C}^*) \leq O_{|Q|}(\log n)$ and $\text{SIZE}(\mathtt{C}) \leq O_k(T_{det}(Q, D_\Omega))$. Using this along with Lemma 4.9, Theorem 4.7 and the fact that $n \leq T_{det}(Q, D_\Omega)$, we have the following corollary:

(3.1) (3.2) Argue that then that overall for the monomial num. frac. of cancellations multiply i.e. small

$\gamma \geq 1 - \prod_{i=1} (c+1)^{-(d_i-1)} \geq 1-(c+1)^{-(k-1)}$

▶ **Corollary 4.10.** *Let Q be an $\mathcal{RA}^+$ query and $\mathcal{D}$ be a Binary-BIDB with $p_0 > 0$ and $\gamma < 1$ (where p_0, γ as in Theorem 4.7) are absolute constants. Let $\Phi(\mathbf{X}) = \Phi[Q, D, t]$ for any result tuple t with $\deg(\Phi) = k$. Then one can compute an approximation satisfying Eq. (3) in time $O_{k,|Q|,\epsilon',\delta}(T_{det}(OPT(Q), D, c))$ (given Q, D and p_i for each $i \in [n]$ that defines $\mathcal{P}$).*

Next, we note that the above result along with Lemma 4.8 answers Problem 1.5 in the affirmative as follows:

Also this goes to appendix?

▶ **Corollary 4.11.** *Let Q be an $\mathcal{RA}^+$ query and $\mathcal{D}$ be a c-TIDB with $p_0 > 0$ (where p_0 as in Theorem 4.7) is an absolute constant. Let $\Phi(\mathbf{X}) = \Phi[Q, D, t]$ for any result tuple t with $\deg(\Phi) = k$. Then one can compute an approximation satisfying Eq. (3) in time $O_{k,|Q|,\epsilon',\delta,c}(T_{det}(OPT(Q), D, c))$ (given Q, D and $p_{t,j}$ for each $t \in D$, $j \in [c]$ that defines $\mathcal{P}$).*

why do you need this?

Proof of Corollary 4.11. The proof follows by Proposition 2.4, Lemma 4.8, and Corollary 4.10. ◀

If we want to approximate the expected multiplicities of all $Z = O(n^k)$ result tuples t simultaneously, we just need to run the above result with δ replaced by $\frac{\delta}{Z}$. Note this increases the runtime by only a logarithmic factor.

5 Related Work

Probabilistic Databases (PDBs) have been studied predominantly for set semantics. Approaches for probabilistic query processing (i.e., computing marginal probabilities of