Pessimistic Cardinality Estimation*

Mahmoud Abo Khamis
RelationalAI, Inc

Kyle Deeds
University of Washington

Dan Olteanu
University of Zurich

Dan Suciu
University of Washington

ABSTRACT

Cardinality Estimation is to estimate the size of the output of a query without computing it, by using only statistics on the input relations. Existing estimators try to return an unbiased estimate of the cardinality: this is notoriously difficult. A new class of estimators have been proposed recently, called *pessimistic* estimators, which compute a guaranteed upper bound on the query output. Two recent advances have made pessimistic estimators practical. The first is the recent observation that degree sequences of the input relations can be used to compute query upper bounds. The second is a long line of theoretical results that have developed the use of information theoretic inequalities for query upper bounds. This paper is a short overview of pessimistic cardinality estimators, contrasting them with traditional estimators.

1. INTRODUCTION

Given a query, the *Cardinality Estimation* problem, or CE for short, is to estimate the size of the query output, using precomputed statistics on the input database. This estimate is the primary metric guiding cost-based query optimization. The optimizer uses it to make decisions about every aspect of query execution. These range from broad logical optimizations like the join order and the number of servers to distribute the data over, to detailed physical optimizations like the use of bitmap filters and memory allocation for hash tables.

Unfortunately, CE is notoriously difficult. Density-based estimators, pioneered by System R [35], are widely used today, but they underestimate significantly due to their strong reliance on assumptions like data uniformity and independence. They tend to have large errors for queries with many joins and many predicates [11, 18]. A major limitation of density-based CE is that it does not come with any theoretical guarantees: it may under-, or over-estimate, by a little or by a lot, without any warning. Alternative approaches have been proposed, but they come with their own limitations: sampling-based estimators require expensive data access at estimation time, while ML-based estimators suffer from large memory footprints and long training time [11,27,37].

An alternative to *estimation* is to compute an *upper bound* on the size of the query output, based on the precomputed statistics on the data. This is called *pessimistic cardinality estimation*, or PCE for short [4]. The PCE returns a number that is larger than the size of the query output for *any* database instance that satisfies the given statistics. The inclusion of the term "estimation" in PCE is a little misleading, since it does not estimate but gives an upper bound, yet the term has stuck in the literature, and we will use it too. Of course, we want the upper bound to be as small as possible, but not smaller. When it achieves accuracy that is similar to, or even better than traditional estimators, PCE offers several advantages. Its one-sided theoretical guarantee can be of use in many applications, for example it can guarantee that a query does not run out of memory, or it can put an upper bound on the number of servers required to distribute the output data. It also has the advantage that the upper bounds combine naturally: if we have multiple upper bounds due to different techniques, we can apply them all and take the minimum.

This paper gives a high level overview of pessimistic cardinality estimation, and contrasts it with traditional methods. We start by discussing single-block SQL queries:

```
SELECT *
FROM R1, R2, ...                    (1)
WHERE [joins and filters]
```

Later we will restrict our discussion to conjunctive queries. Some of the PCE techniques described here also apply to `group-by` queries.

*This work was partially supported by NSF-BSF 2109922, NSF IIS 2314527, NSF SHF 2312195, SNSF 200021-231956, and RelationalAI.

2. BRIEF REVIEW OF CE

Modern database engines use a *density-based* approach for cardinality estimation. They compute periodically some simple statistics (or summaries) of the base relations, then use simplifying assumptions to estimate the cardinality. Specifically, the engine stores in its catalog, for each relation R, its cardinality $|R|$, and the number of distinct values[1] $|\mathsf{Dom}(R.X)|$, where X is a single attribute or a set of attributes. This quantity is computed periodically, and approximatively, by sampling from R (e.g. Postgres) or by using Hyper-Log-Log (e.g. DuckDB). The ratio $\frac{|R|}{|\mathsf{Dom}(R.X)|}$ represents the *average degree*, or the *density* of the attribute(s) X.

Consider the uniform probability space whose outcomes are the tuples of the Cartesian product of all relations in (1), $R_1 \times R_2 \times \ldots$ This defines a probability $\Pr(X, Y, \ldots)$ over all attributes returned by the query. A *density-based* CE estimates the probability that a random tuple from the Cartesian product satisfies the condition in the `where` clause. The cardinality estimate is the multiplication of this probability with the size of the Cartesian product. For example, consider the following SQL query Q:

```
SELECT * FROM Store WHERE City = 'Seattle'
```

By the *uniformity assumption* and the *containment of values assumption* the probability that a random tuple satifies the predicate is $\frac{1}{|\mathsf{Dom}(\texttt{Store.City})|}$. The cardinality estimate is $\textsc{Est}(Q) = \frac{|\texttt{Store}|}{|\mathsf{Dom}(\texttt{Store.City})|}$.

1-Dimensional Histograms relax the uniformity assumption, by storing separate statistics for each bucket of a histogram. The default number of buckets is small (200 for SQLServer [1], 100 for Postgres), and is strictly limited (typically to $1000-10000$ [37]).

Joins are estimated using similar assumptions. For example, consider the query J given by:

```
SELECT * FROM R, S WHERE R.X = S.Y
```  (2)

One estimate could be $|R| \cdot \frac{|S|}{|\mathsf{Dom}(S.Y)|}$, because each tuple in R matches an estimated $\frac{|S|}{|\mathsf{Dom}(S.Y)|}$ tuples in S. Or, symmetrically, $\frac{|R|}{|\mathsf{Dom}(R.X)|} \cdot |S|$. Density-based CE returns their minimum[2], usually written as:

$$\textsc{Est}(J) = \frac{|R| \cdot |S|}{\max(|\mathsf{Dom}(R.X)|, |\mathsf{Dom}(S.Y)|)} \quad (3)$$

Finally, the estimate for a conjunction of predicates is computed by assuming independence between the predicates. For example, the estimate of the following query Q

```
SELECT * FROM Store
WHERE City = 'Seattle' AND Zip = '98195'
```

is $\textsc{Est}(Q) = \frac{|\texttt{Store}|}{|\mathsf{Dom}(\texttt{Store.City})| \cdot |\mathsf{Dom}(\texttt{Store.Zip})|}$, which is an underestimate, since that entire zip code is in Seattle. *2-Dimensional Histograms* can capture correlations between attributes, but few systems support them.[3]

A landmark paper [29] evaluated the cardinality estimators deployed in modern database systems and their impact on the query optimizer. It found that their CE almost always underestimates (because of the independence assumption) with typical errors of up to 10^4 for queries with many joins. Later studies have confirmed these findings across a wide variety of workloads and database systems [18, 27, 28, 34, 37].

Given the importance of the problem and the limitations of density-based methods, many recent proposals have been published exploring alternatives to traditional density-based CE. Two alternatives have attracted particular interest: *sampling-based* CE and *learned* CE.

Sampling-based CE methods compute an unbiased estimate without requiring any assumptions. *Offline sampling* pre-computes a uniform sample $R_{\text{sample}} \subseteq R$ of each relation, then estimates the size of the query output over the base relations from the size of the query output over the sample using the Horvitz-Thompson's formula.[4] This method can be very accurate for queries over a single relation and easily supports arbitrary user-defined predicates beyond equality and range predicates. However, it becomes ineffective for highly selective predicates, because of *sampling collapse*: when no sampled tuple matches the query, then the system must return 0 or 1. In particular, this is a problem for joins, since their selectivity is relative to the cartesian product of the input relations and therefore almost always extremely low.[5] *Online sampling* addresses this issue by sampling only tuples that join with already

[1] Denoted $V(R, X)$ in a popular textbook [14].

[2] Justified by the *containment-of-values* assumption.

[3] There are several reasons why 2-d histograms are rarely used. (1) There are too many candidates: $n(n-1)/2$ possible histograms for a relation with n attributes. The number of buckets along each dimension is limited to $\sqrt{1000} - \sqrt{10000}$. (2) It is unclear how to combine multiple 2-d histograms [32], e.g. in order to estimate a predicate on 3 attributes $\Pr(X, Y, Z)$ from 2-d histograms on XY, XZ, YZ.

[4] The size estimate is the multiplication of the size of the query output over the sample with the ratio $|R|/|R_{\text{sample}}|$ for each relation in the query. More robust estimates, such as *bottom-k* [7], are not commonly used for CE [6].

[5] The estimate is still unbiased, over the random choices of the samples, but the standard deviation is high.

sampled tuples. Based on this principle, Wander Join [30, 31] achieves remarkable accuracy [34]; however, it requires access to an index on every join column, and has a high latency.

Learned CE aims to remove the assumptions of traditional CE by training an ML model that captures the complex correlations empirically [27, 37]. *Data-driven* estimators compute a generative ML model for the probability $\Pr(X, Y, \ldots)$ over all attributes returned by the query. This is trained on the database [38, 39]. The model needs to represent *all* attributes, of *all* relations. Intuitively, these models aim for a lossy compression of the full outer join of the database relations then estimate the selectivity of the predicates in the query relative to it. This is an extremely ambitious approach, and it tends to require large models that either struggle with, or completely disallow queries that do not follow the schema's natural join structure, like self-joins. *Query-driven estimators* compute a discriminative model for $\textsc{Est}(Q)$. The model is trained using a workload of queries and their true cardinalities. In general, ML-based estimates can be quite accurate on the training data, but they suffer from distribution drift, can be memory intensive (1MB to 1GB models are reported in the literature), support only limited types of queries and predicates, and require full retraining when the data changes, e.g. when a new relation is added [18].

3. A WISH LIST

Stepping back from existing estimators, we ask a more general question: What properties do we wish a cardinality estimator to have? We propose here six such properties, which we argue any good CE should have:

Accuracy/Speed/Memory: It should have good accuracy, small estimation time, small memory footprint.

Locality: It should use statistics that are computed separately on each input relation.

Composition: It should be able to compute an estimate for a query from the estimates for its subqueries; this is useful in bottom-up query optimizers.

Combination: It should be able to combine multiple sources of statistics on the database. Given two estimates $\textsc{Est}_1$ and $\textsc{Est}_2$ computed using different methods, or different statistics, one should be able to combine them to obtain a better estimate $\textsc{Est}$.

Incremental Updates: It should be possible to update the statistics incrementally when the input data is updated.

Guarantees: It should offer some theoretical guarantees. This will allow the application to reason about decisions based on CE.

All CE's aim for good speed/accuracy/memory, with various degrees of success. Density-based and sampling-based CE's are local, while learned CE are definitely not local: they are monolithic, in that they require access to all relations at training time. Density-based CE is compositional[6], but it cannot do combination. For example, to estimate the size of $\sigma_{X=a \wedge Y=b \wedge Z=c}(R)$ we could use $|\mathsf{Dom}(R.X)|$ and $|\mathsf{Dom}(R.YZ)|$, or we could use $|\mathsf{Dom}(R.XY)|$ and $|\mathsf{Dom}(R.Z)|$, but we cannot combine these two estimates to get a better estimate [32]. For another example, given the cardinalities of both joins $R \bowtie S$ and $S \bowtie T$, there is no canonical way to combine them to estimate the cardinality of $R \bowtie S \bowtie T$ [5]. ML-based estimators can be neither composed nor combined, and need to be re-trained from scratch after an update. Finally, of all methods discussed here, only sampling-based methods offer theoretical guarantees: the others offer no guarantees.

Next, we will discuss the alternative, pessimistic approach to cardinality estimation. We will return to our wish-list in Sec. 11.

4. PESSIMISTIC CE

A *Pessimistic Cardinality Estimator*, PCE, computes a *guaranteed upper bound* on the cardinality, instead of an estimate. For a very simple example, an upper bound of the join (2) is $|R| \cdot |S|$. If we know the largest number b of occurrences of any value in $S.Y$, also called the *maximum degree of* Y *in* S, then a better bound is $|R| \cdot b$; for example, if Y is a key in S, then $b = 1$ and the bound becomes $|R|$. Symmetrically, if we know the maximum degree of Y in R, call it a, then $a \cdot |S|$ is also an upper bound. We can combine these bounds by taking their min:

$$|J| \leq \min(|R| \cdot b,\ a \cdot |S|) \tag{4}$$

This should be compared with the traditional estimator (3), which replaces the maximum degrees a and b with the average degrees.

Two advances make pessimistic cardinality estimators practical today. The first is the observation that degree sequences can be used to compute an upper bound on the query output size [10]. The second is a long line of theoretical results on using

[6]Under the *preservation of values* assumption.

$R =$

X	Y	Z
1	a	...
1	b	...
1	b	...
2	a	...
2	b	...
3	b	...
3	c	...
4	d	...

$\mathtt{deg}_R(YZ|X) = (3,2,2,1)$
$\mathtt{deg}_R(Y|X) = (2,2,2,1)$
$\mathtt{deg}_R(Z|XY) = (2,1,1,1,1,1,1)$
$\mathtt{deg}_R(XYZ|\emptyset) = (8) = (|R|)$

Figure 1: Example of Degree Sequences

information inequalities to compute upper bounds on the query's output [13, 15–17, 23, 25, 26, 36]. We review both topics in the rest of the paper.

5. DEGREE SEQUENCES

The existing PCE's use statistics derived from *degree sequences* of the input relations.

Let R be a relation, and $U, V \subseteq \mathtt{Attrs}(R)$ two sets of attributes. We assume throughout the paper that relations are *sets*. The *degree sequence*

$$\mathtt{deg}_R(V|U) \stackrel{\text{def}}{=} (d_1, d_2, \ldots, d_n) \tag{5}$$

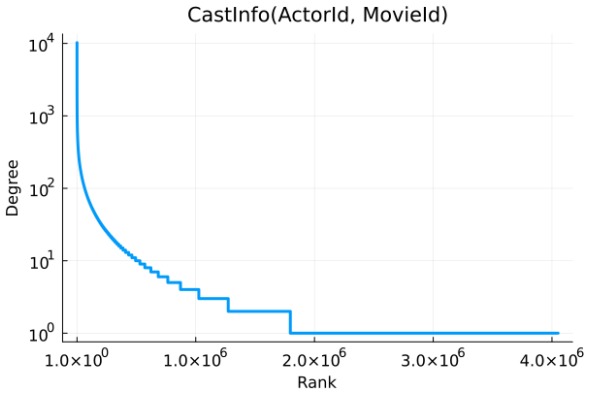
is defined as follows. If $u_1, \ldots, u_n$ are the distinct values of $\Pi_U(R)$, then $d_i \stackrel{\text{def}}{=} |\sigma_{U=u_i}(\Pi_{UV}(R))|$ is the degree of the value u_i. We assume that the values u_i are sorted in decreasing order of their degrees, i.e. $d_1 \geq d_2 \geq \cdots \geq d_n$. We say that the value u_i has *rank* i. Notice that $\mathtt{deg}_R(V|U) = \mathtt{deg}_R(UV|U)$.

Figure 1 illustrates some simple examples of degree sequences, while Figure 2 shows the degree sequence of the IMDB dataset from the JOB benchmark [29]. If the functional dependency $U \to V$ holds, then $\mathtt{deg}_R(V|U) = (1, 1, \ldots, 1)$. When $|U| \leq 1$ then we call the degree sequence $\mathtt{deg}_R(V|U)$ *simple*, and when $UV = \mathtt{Attrs}(R)$ are all the attributes of R, then we call $\mathtt{deg}_R(V|U)$ *full* and also denote it by $\mathtt{deg}_R(*|U)$. The degree sequence $\mathtt{deg}_R(YZ|X)$ in Fig. 1 is both simple and full, and we can write it as $\mathtt{deg}_R(*|X)$.

The ℓ_p*-norm* of the degree sequence (5) is:

$$||\mathtt{deg}_R(V|U)||_p \stackrel{\text{def}}{=} (d_1^p + d_2^p + \cdots + d_n^p)^{1/p}$$

Degree sequences and their ℓ_p-norms generalize common statistics used by density-based estimators: the relation cardinality is $||\mathtt{deg}_R(*|U)||_1 = |R|$, the number of distinct values $|\mathsf{Dom}(R.U)|$ is the length n of the sequence (5) (and also equal to $||\mathtt{deg}_R(U|\emptyset)||_1$); and the maximum degree of $R.U$ is $||\mathtt{deg}_R(*|U)||_\infty = d_1$. But degree sequences contain information that can significantly improve the accuracy of PCE. For example the inequality $|J| \leq a \cdot |S|$ in (4) assumes pessimistically that all degrees in $\mathtt{deg}_R(X|Y)$ are equal to the maximum degree a. By using the degree



$||\mathtt{deg}_{\text{CastInfo}}(\text{MovieId}|\text{ActorID})||_1 = 36 \cdot 10^6$
$||\mathtt{deg}_{\text{CastInfo}}(\text{MovieId}|\text{ActorID})||_\infty = 10^4$

Figure 2: The degree sequence of CastInfo from the JOB benchmark [29]. Its cardinality is $36 \cdot 10^6$, and it has $4 \cdot 10^6$ distinct actor IDs, with degrees ranging from 10^4 to 1. The maximum degree is that of Bob Barker, who hosted the CBS show *The Price Is Right* from 1972 to 2007 and also *Truth or Consequences* from 1956 to 1975.

sequence and more sophisticated inequalities we can account for the true distribution of the data, as we will see later.

In the rest of the paper we describe several methods that compute an upper bound on the size of the query output from the ℓ_p-norms of degree sequences. We restrict our discussion to conjunctive queries instead of the SQL query (1) and use the notation:

$$Q(X_1, \ldots, X_n) = R_1(U_1) \wedge \ldots \wedge R_m(U_m) \tag{6}$$

where $\{X_1, \ldots, X_n\} = U_1 \cup \cdots \cup U_m$. The *query variables* are $X_1, \ldots, X_n$. We omit filter predicates for now, and discuss them in Sec. 10.

6. THE AGM BOUND

Historically, the first cardinality upper bound was introduced by Atserias, Grohe, and Marx [3], and builds on earlier work by Friedugut and Kahn [13] and Grohe and Marx [16]. The bound is known today as the AGM bound. It uses only the cardinalities of the input relations, $|R_1|, \ldots, |R_m|$, and is defined as follows. A *fractional edge cover* of the query Q in (6) is a tuple of non-negative weights, $\boldsymbol{w} = (w_1, w_2, \ldots, w_m)$, such that every variable X_j is "covered", meaning $\sum_{i:X_j \in U_i} w_i \geq 1$, for $1 \leq j \leq n$. Then, the following inequality holds:

$$|Q| \leq |R_1|^{w_1} \cdot |R_2|^{w_2} \cdots |R_m|^{w_m} \tag{7}$$

The AGM bound of Q is defined as the minimum of (7), taken over all fractional edge covers $\boldsymbol{w}$.

The classical illustration of the AGM bound is for the 3-cycle query:

$$C_3(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,X) \quad (8)$$

One fractional edge cover is $(1,1,0)$, which proves $|C_3| \leq |R| \cdot |S|$. Other fractional edge covers are $(1,0,1)$, $(0,1,1)$, and $(\frac{1}{2},\frac{1}{2},\frac{1}{2})$, and each leads to a similar upper bound on $|C_3|$. The AGM bound is their minimum:[7]

$$|C_3| \leq \min\left(|R|\cdot|S|,\ |R|\cdot|T|,\ |S|\cdot|T|,\ (|R|\cdot|S|\cdot|T|)^{1/2}\right)$$

The AGM bound enjoys two elegant properties: it is computable in PTIME in the size of the query Q (by solving a linear program), and the bound is guaranteed to be tight. The latter means that, for any set of cardinalities $|R_1|, |R_2|, \ldots$ there exists a *worst case database instance*, with the same cardinalities, where the size of the query output is equal to the AGM bound, up to a rounding error equal to a query-dependent multiplicative constant.[8]

Despite these attractive theoretical properties, the AGM bound has not been adopted in practice, because it uses very limited statistics on the input data, namely just the relation cardinalities. In particular, for acyclic queries the AGM bound is achieved by an integral (not fractional) edge cover, and expression (7) is a product of the cardinalities of some relations. For example, for the 2-way join query:

$$J_2(X,Y,Z) = R(X,Y) \wedge S(Y,Z) \quad (9)$$

The AGM bound is $|R| \cdot |S|$. In contrast, a density-based estimator uses both cardinalities and average degrees, as seen for example in (3), and for this reason it returns a better estimate for J_2 than the AGM bound, even if it cannot provide any guarantees.

7. THE CHAIN BOUND

The chain bound [25], CB for short, uses as statistics both cardinalities $|R_i|$, and maximum degrees $||\mathtt{deg}_{R_i}(V|U)||_\infty$. It strictly generalizes the AGM bound, but a popular simplification adopted by several implementations is weaker than the AGM bound. To describe CB, it is convenient to view each cardinality $|R_j|$ as a maximum degree, namely $|R_j| = ||\mathtt{deg}_{R_j}(*|\emptyset)||_\infty$. Thus, all statistics are max degrees: $||\mathtt{deg}_{R_{j_i}}(V_i|U_i)||_\infty$, for $i = 1, k$.

Fix an ordering π of the query variables. We say that the pair $(V|U)$ *covers* the variable X w.r.t. π, and write $X \in_\pi (V|U)$, if $X \in V$ and all variables in U strictly precede X in the order π. For a simple example, if $U = \emptyset$, then $(V|\emptyset)$ covers all variables in V; for another example, if the order π is X, Y, Z then $(XZ|Y)$ covers only Z. If a set of non-negative weights $w_1, \ldots, w_k$ is a *fractional cover* of $(V_1|U_1), \ldots, (V_k|U_k)$, meaning that it covers every variable X (i.e., $\sum_{i:(X \in_\pi V_i|U_i)} w_i \geq 1$), then the following holds (we give the proof in the Appendix):

$$|Q| \leq \prod_{i=1,k} ||\mathtt{deg}_{R_{j_i}}(V_i|U_i)||_\infty^{w_i} \quad (10)$$

CB is the minimum of the quantity above over all fractional covers $\boldsymbol{w}$ and variable orderings π, and is always an upper bound on $|Q|$. If π is fixed, then CB can be computed in PTIME by using an LP solver, but minimizing over all π requires exponential time.[9] When all statistics are cardinalities then CB is independent on the choice of π and coincides with the AGM bound. When the statistics also include max-degrees, then CB is lower (better) than the AGM bound.

Several implementations have adopted a simplified version of CB which does not require the use of an LP solver, called BoundSketch in [4, 19] or MOLP in [5]: we will use the term BoundSketch. It corresponds to restricting the fractional cover $\boldsymbol{w}$ in (10) to be an integral cover.[10] We describe BoundSketch, following the presentation in [5]. Consider the following nondeterministic algorithm.

- Set $W := \emptyset$, $CB := 1$.
- Repeatedly choose non-deterministically one statistics, $d_i \stackrel{\text{def}}{=} ||\mathtt{deg}_{R_{j_i}}(V_i|U_i)||_\infty$, such that $U_i \subseteq W$. Set $W := W \cup V_i$ and $CB := CB \cdot d_i$.
- Stop when W contains all query variables, and return CB.

BoundSketch is the minimum value returned by all executions of the non-deterministic algorithm. This is equivalent to computing the shortest path in the graph called CEG_M in [5]: its nodes are all subsets $W \subseteq \{X_1, \ldots, X_n\}$, and for every statistics $d_i = ||\mathtt{deg}_{R_{j_i}}(V_i|U_i)||_\infty$ there is an edge with weight d_i from W to $W \cup V_i$, for all $W \supseteq U_i$. Then BoundSketch is the shortest[11] path from $\emptyset$ to $\{X_1, \ldots, X_n\}$. BoundSketch still requires exponential time.[12]

[7] These four fractional edge covers are called the *vertices of the edge cover polytope*. In (7) it suffices to restrict the min to the vertices of the edge cover polytope, because any other fractional edge cover is $\geq$ than some convex combination of the vertices.

[8] The constant is $\frac{1}{2^n}$, where n is the number of variables.

[9] Computing CB is NP-hard [20].

[10] BoundSketch also hash partitions the data in order to improve the estimate [4]. We will use BoundSketch to refer to integral CB, without data partitioning.

[11] Weights along a path are multiplied.

[12] When the statistics are restricted to cardinalities (no

BoundSketch is similar to a density-based estimator, where the average degrees are replaced by maximum degrees: this was noted in [5]. We illustrate this on an example with a 3-way join query:

$$J_3(X,Y,Z,U) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,U) \quad (11)$$

BoundSketch is the minimum of the following quantities:[13]

$$\begin{aligned}
|J_3| &\le |R| \cdot ||\mathtt{deg}_S(Z|Y)||_\infty \cdot ||\mathtt{deg}_T(U|Z)||_\infty \quad (12)\\
|J_3| &\le ||\mathtt{deg}_R(X|Y)||_\infty \cdot |S| \cdot ||\mathtt{deg}_T(U|Z)||_\infty \\
|J_3| &\le ||\mathtt{deg}_R(X|Y)||_\infty \cdot ||\mathtt{deg}_S(Y|Z)||_\infty \cdot |T| \\
|J_3| &\le |R| \cdot |T|
\end{aligned}$$

Let us compare this with the density-based estimator described in Sec. 2, which, for J_3, is:

$$\textsc{Est}(J_3) = \frac{|R| \cdot |S| \cdot |T|}{\max(|R.Y|, |S.Y|) \cdot \max(|S.Z|, |T.Z|)}$$

For illustration, assume that $|R.Y| \le |S.Y|$ and $|S.Z| \le |T.Z|$. Then the estimate can be written as:

$$\textsc{Est}(J_3) = |R| \cdot \frac{|S|}{|S.Y|} \cdot \frac{|T|}{|T.Z|}$$

This is the same expression as (12), where the maximum degrees are replaced with the average degrees.

The main advantage of CB and its simplified variant BoundSketch is their simplicity and resemblance to the density-based estimator. A disadvantage is that their computation requires exponential time, because we need to iterate over all variable orderings π. Another limitation is that, unlike the AGM bound, CB is not tight: the polymatroid bound described below can be strictly lower than CB. However, in the special case when the set of statistics is *acyclic*, CB is both tight and computable in PTIME. In this case, there exist variable orderings such that, for every statistics $||\mathtt{deg}_{R_{j_i}}(V_i|U_i)||_\infty$, all variables in U_i come before those in V_i, which implies $V_i \subseteq_\pi (V_i|U_i)$ for $i = 1, k$. Such an ordering π can be computed in linear time using topological sort. We then use an LP solver to compute the optimal fractional cover for π.

8. THE POLYMATROID BOUND

The *Polymatroid Bound* [26], PolyB, can use any ℓ_p-norms of degree sequences to compute an upper bound on the size of the query output. The input statistics for PolyB are norms $||\mathtt{deg}_R(V|U)||_p$, for a relation R, sets of attributes U, V of R, and numbers[14] $p > 0$. The system uses all available statistics to compute an upper bound on the query's output. PolyB strictly generalizes both CB and the AGM bound. Its theoretical foundation lies in information inequalities, which we will review shortly, after presenting some examples of PolyB.

Examples To warm up, consider again the two-way join $J_2(X,Y,Z) = R(X,Y) \wedge S(Y,Z)$. The following upper bounds on the size of the query output hold:

$$\begin{aligned}
|J_2| &\le |R| \cdot |S| \quad (13)\\
|J_2| &\le |R| \cdot ||\mathtt{deg}_S(Z|Y)||_\infty \quad (14)\\
|J_2| &\le ||\mathtt{deg}_R(X|Y)||_\infty \cdot |S| \quad (15)\\
|J_2| &\le ||\mathtt{deg}_R(X|Y)||_2 \cdot ||\mathtt{deg}_S(Z|Y)||_2 \quad (16)
\end{aligned}$$

The PolyB is the minimum of these four quantities[15]. The first three inequalities are trivial (and were discussed in Sec. 4), while (16) follows from Cauchy-Schwartz.

Consider now the 3-way join $J_3(X,Y,Z,U) = R(X,Y) \wedge S(Y,Z) \wedge T(Z,U)$. PolyB includes all CB inequalities (see Eq. (12)) and new inequalities using ℓ_p-norms, for example the following inequality holds for all $p \ge 2$:

$$\begin{aligned}
|J_3| \le \big(&|R|^{p-2} \cdot ||\mathtt{deg}_R(X|Y)||_2^2 \\
&\cdot ||\mathtt{deg}_S(Z|Y)||_{p-1}^{p-1} \cdot ||\mathtt{deg}_T(U|Z)||_p^p\big)^{1/p} \quad (17)
\end{aligned}$$

For example, assume that the system has computed the ℓ_1, ℓ_2, ℓ_3 norms of all degree sequences, then it can compute the expression above for both $p = 2$ and $p = 3$, and take their minimum.

Considering the 3-cycle query in (8), the following inequalities hold:

$$\begin{aligned}
|C_3| &\le (|R| \cdot |S| \cdot |T|)^{1/2} \quad (18)\\
|C_3| &\le \big(||\mathtt{deg}_R(Y|X)||_2^2 \cdot ||\mathtt{deg}_S(Z|Y)||_2^2 \cdot ||\mathtt{deg}_T(X|Z)||_2^2\big)^{1/3} \quad (19)\\
|C_3| &\le \big(||\mathtt{deg}_R(Y|X)||_3^3 \cdot ||\mathtt{deg}_S(Y|Z)||_3^3 \cdot |T|^5\big)^{1/6} \quad (20)
\end{aligned}$$

The first is the AGM bound; the others are new, and quite surprising. This list is not exhaustive: one can derive many more inequalities, using various ℓ_p-norms. PolyB is the minimum of all of them (restricted to the available ℓ_p-norms), and can be computed using a linear program, as explained later.

max-degrees), then BoundSketch or, equivalently, MOLP [5], is larger (worse) than the AGM bound, as it considers only integral covers. The original definition of MOLP [22] (called MO in [22]) claims the opposite, a claim repeated in [5]. However, that claim only holds under the assumption that one first regularizes the data, then computes a separate bound for each degree configuration, using both cardinalities and max-degree statistics.

[13] We omit non-optimal expressions, like $|R| \cdot |S| \cdot |T|$.

[14] p can be fractional.

[15] If all statistics consists of ℓ_p-norms with p an integer, or $p = \infty$, then the best bound on J_2 is the minimum of inequalities (14)-(16).

Inequalities (17)-(20) do not appear to have simple, elementary proofs. Instead, they are proven using Shannon inequalities.

Shannon Inequalities We review briefly Shannon Inequalities, which are special cases of entropic inequalities, or information inequalities, and show how to use them to prove the inequalities above.

Let X be a finite random variable, with N outcomes $x_1, \ldots, x_N$, and probability function Pr. Its *entropy* is defined as:[16]

$$h(X) \stackrel{\text{def}}{=} -\sum_i \Pr(x_i) \log \Pr(x_i)$$

It holds that $0 \leq h(X) \leq \log N$, and $h(X) = \log N$ iff Pr is uniform, i.e., $\Pr(x_1) = \cdots = \Pr(x_N) = 1/N$.

Let $X_1, \ldots, X_n$ be n finite, jointly distributed random variables. For every subset U of variables, $h(U)$ denotes the entropy of the joint random variables in U. For example, we have $h(X_1X_3)$, $h(X_2X_4X_5)$, etc. This defines an *entropic vector*, $\boldsymbol{h}$, with 2^n dimensions, one for each subset of variables. The following inequalities hold and are called *Shannon basic inequalities* (the second is called *monotonicity* and the third is called *submodularity*):

$$\begin{aligned} h(\emptyset) &= 0 \\ h(U \cup V) &\geq h(U) \\ h(U) + h(V) &\geq h(U \cup V) + h(U \cap V) \end{aligned}$$

A vector $\boldsymbol{h} \in \mathbb{R}_+^{2^n}$ that satisfies Shannon basic inequalities is called a *polymatroid*. Every entropic vector is a polymatroid, but the converse does not hold in general [42].

We now have the tools needed to prove inequalities (17)-(20). We illustrate only (18), and defer the others to the Appendix. Consider three input relations R, S, T, and let C_3 denote the output to the query, i.e. $C_3(x, y, z)$ iff $R(x, y)$, $S(y, z)$, and $T(z, x)$. Define the uniform probability space on C_3: there are $|C_3|$ outcomes (x, y, z), each with the same probability $1/|C_3|$. Let $\boldsymbol{h} \in \mathbb{R}_+^8$ be its entropic vector. Note that $h(XYZ) = \log |C_3|$, because the probability distribution is uniform, and $h(XY) \leq \log |R|$, because the support of the variables XY is a subset of the relation R. Then the following inequalities hold, and imply (18):

$$\begin{aligned} \log|R| + \log|S| + \log|T| \geq \\ \geq & \underline{h(XY) + h(YZ)} + h(ZX) \\ \geq & h(XYZ) + \underline{h(Y) + h(ZX)} \\ \geq & 2h(XYZ) = 2\log|C_3| \end{aligned}$$

We have applied Shannon submodularity inequality twice, and shown it by underlying the used terms.

Entropic Constraints To prove upper bounds that involve degree sequences, like (17), (19), (20), we need to relate their ℓ_p-norms to the entropic vector.

Let $X_1, \ldots, X_n$ be jointly distributed random variables, whose support is the finite relation $R(X_1, \ldots, X_n)$. Let two sets of variables U, V. Let $\boldsymbol{h}$ be the entropic vector of the variables. The *conditional entropy* of U, V is defined as $h(V|U) \stackrel{\text{def}}{=} h(UV) - h(U)$.

The following was proven in [24], for any $p > 0$:

$$\frac{1}{p} h(U) + h(V|U) \leq \log ||\texttt{deg}_R(V|U)||_p \qquad (21)$$

When $p = 1$, the inequality becomes $h(UV) \leq \log |\Pi_{UV}(R)|$, and when $p = \infty$, it becomes $h(Y|X) \leq \log ||\texttt{deg}_R(V|U)||_\infty$. For a simple example, using (21) we can prove (16):

$$\begin{aligned} &\log||\texttt{deg}_R(X|Y)||_2 + \log||\texttt{deg}_S(Z|Y)||_2 \geq \\ &\geq \left(\frac{1}{2}h(Y) + h(X|Y)\right) + \left(\frac{1}{2}h(Y) + h(Z|Y)\right) \\ &= h(Y) + h(X|Y) + h(Z|Y) \\ &\geq h(Y) + h(X|Y) + h(Z|XY) = h(XYZ) = \log|J_2| \end{aligned}$$

Inequalities (17)-(20) are proven similarly: we include them in Appendix B.

Computing PolyB So far, we introduced PolyB as the minimum bound that can be obtained from Shannon inequalities and entropic constraints (21). To compute it, we use an equivalent, dual definition, as the maximum value of a linear program (LP).

Suppose Q has n variables $X_1, \ldots, X_n$. The LP has 2^n real-valued variables, $h(U)$, representing the unknown polymatroid vector $\boldsymbol{h} \in \mathbb{R}_+^{2^n}$. The objective is to maximize $h(X_1X_2 \cdots X_n)$, given two sets of linear constraints:

- Statistics constraints: there is one inequality of type (21) for each available input statistics.
- Shannon basic inequalities: the list of all submodularity and monotonicity inequalities.

PolyB is the optimal value of this linear program. Intuitively, we maximize $\log|Q| = h(X_1 \cdots X_n)$, while constraining $\boldsymbol{h}$ to be a polymatroid that satisfies all statistics (21).

We illustrate with an example, by showing the linear program for the query C_3 in (8):

[16] Usually log is in base 2, but any base can be used without affecting any results.

maximize $h(XYZ)$ subject to		
$h(XY) \leq$	$\log \lvert R \rvert$	cardinality
$h(XZ) \leq$	$\log \lvert S \rvert$	constraints
...		
$\frac{1}{2}h(X) + h(Y\lvert X) \leq$	$\log \lVert \mathtt{deg}_R(Y\lvert X) \rVert_2$	Other ℓ_p
$\frac{1}{3}h(X) + h(Y\lvert X) \leq$	$\log \lVert \mathtt{deg}_R(Y\lvert X) \rVert_3$	constraints
...		
$h(X) + h(Y) \geq$	$h(XY)$	all Shannon
$h(XY) + h(XZ) \geq$	$h(X) + h(XYZ)$	inequalities
...		

PolyB has several attractive properties. It strictly generalizes both CB and the AGM bound, which use only statistics given by ℓ_1 and ℓ_∞. In addition, PolyB can also be used to estimate output sizes for queries with group-by (and select distinct) clauses; for this, we just need to adjust the objective of the linear program to refer to the entropic term for the group-by attributes. Access to more statistics is guaranteed to never make the bound worse; e.g., we can improve the bound if we decide to add the statistics ℓ_3, ℓ_4, ℓ_5 in addition to $\ell_1, \ell_2, \ell_\infty$. The inference time is exponential in n, the number of query variables, but several special cases are known when it can be computed in PTIME [21, 33, 41] (see Sec. 10). Finally, when all statistics are restricted to *simple* degree sequences, then PolyB is provably tight, up to a query-dependent multiplicative rounding error,[17] similar to the AGM bound. Beyond simple degree sequences, the PolyB is not tight in general, due to the existence of non-Shannon inequalities [36, 42], but the only known non-tight examples are artificial and unlikely to occur in practice.

9. THE DEGREE SEQUENCE BOUND

The *Degree Sequence Bound* [10, 11], DSB for short, was the first system that proposed to use degree sequences for PCE. DSB provides tighter bounds than both AGM and CB bounds, and an empirical evaluation [11] found it to be often tighter than density-based estimators, while always returning a guaranteed upper bound. Instead of using ℓ_p-norms, DSB uses compressed representation of the degree sequences. We illustrate here DSB assuming access to the full degree sequence, and discuss compression in Sec. 10.

Consider the 2-way join $J_2(X, Y, Z) = R(X, Y) \wedge S(Y, Z)$, and assume that the degree sequences of $R.Y$ and $S.Y$ are:

$$\mathtt{deg}_R(*|Y) = (a_1, a_2, \ldots) \quad \mathtt{deg}_S(*|Y) = (b_1, b_2, \ldots)$$

[17] The factor is $1/2^{2^n-1}$.

Recall that the degree sequences are sorted, e.g. $a_1 \geq a_2 \geq \cdots$, and that the rank of a value y in $R.Y$ is the index i for which the degree of y is a_i. If every value y has the same rank in $R.Y$ and in $S.Y$, then the size of the join is precisely $|J_2| = \sum_i a_i b_i$. In general, the following inequality holds:

$$|J_2| \leq \sum_i a_i b_i \tag{22}$$

This bound is a strict improvement over the chain bound (4), because $\sum_i a_i b_i \leq a_1 \sum_i b_i = a_1 |S|$, and similarly $\sum_i a_i b_i \leq |R| b_1$.

DSB enjoys several nice properties. It can be computed in linear time in the size of the compressed degree sequences; it is compositional, meaning that the estimate of a query plan can be computed bottom-up; given appropriate histograms (see Sec. 10) DSB is more accurate than density-based estimators [11]; finally, DSB is provably a tight upper bound of the query's output. DSB also has a few limitations. It is limited to Berge-acylic queries (see [12]), and its estimate is not given in terms of an inequality, like PolyB, but instead it is computed by an algorithm.

10. PRAGMATIC CONSIDERATIONS

We discuss here some practical aspects that need to be considered by PCE, or have been considered in previous implementations of PCE [4, 11, 19, 41].

Statistics selection While a density-based estimator only stores the number of distinct values $|\mathsf{Dom}(R.X)|$ of an attribute, PolyB can use multiple ℓ_p-norms of various degree sequences $\mathtt{deg}_R(V|U)$. A reasonable choice is to store, say, four numbers $\ell_2, \ell_3, \ell_4, \ell_\infty$. We assume that $|R|$ is stored separately and do not include ℓ_1. This means that the memory footprint increased by 4 times that of a density-based estimator. But, in general, it is not clear whether we want to stop at ℓ_4. An implementation needs to choose which ℓ_p norms to precompute. Empirically, we have observed improvements of accuracy up to ℓ_{30} for JOB queries on the IMDB dataset, although with diminishing returns [41]. On the other hand, DSB compresses the degree sequence, and requires tuning the hyperparameters of the compression.

Computing the statistics All input statistics are computed offline. For example, the degree sequence $\mathtt{deg}_R(*|X)$ can be computed using a group-by query:

```
SELECT X, COUNT(*) FROM R GROUP BY X
```
(23)

DSB sorts the query output, while PolyB computes all desired ℓ_p-norms in one pass over this output.

Conditional statistics In order to improve the estimates of queries with predicates, a pessimistic estimator can adopt traditional techniques like Most

Common Values (MCV) and histograms. Assume that the system decides to store the following *global statistics* for column X: $||\texttt{deg}_R(*|X)||_p$ for $p = 2, 3, 4, \infty$, for a total of 4 numbers (recall that ℓ_1 is the cardinality which we store anyway). To support equality predicates $A = a$ on some attribute A, the system computes additional *conditional statistics*:

- For each value $a \in \Pi_A(R)$, compute the degree sequence $\texttt{deg}_{\sigma_{A=a}(R)}(*|X)$, which we denote by $\texttt{deg}_R(*|X, A = a)$:

  $$\texttt{CREATE TABLE } DS \texttt{ AS SELECT } A, X, \quad (24)$$
  $$\texttt{COUNT(*) as } C \texttt{ FROM } R \texttt{ GROUP BY } A, X;$$

  Then, compute the ℓ_p-norms of these degree sequences (shown only for ℓ_2 below):

  $$\texttt{SELECT } A, (\texttt{SQRT}(\texttt{SUM}(C * C))) \texttt{ AS } L_2 \quad (25)$$
  $$\texttt{FROM } DS \texttt{ GROUP BY } A;$$

- For each Most Common Value (MCV) a, store its ℓ_p-norms in the catalog. Call these *conditional MCV statistics*. If we used 100 MCVs (which is the default in Postgre), then these statistics consist of 400 numbers.

- Compute the *conditional common statistics*, $\max_a ||\texttt{deg}_R(*|X, A = a)||_p$, and add these 4 numbers to the catalog.

- At estimation time, consider three cases. If the query does not have a predicate on A, then use the global statistics; if the predicate is $A = a$, where a is an MCV, then use the corresponding conditional MCV statistics; otherwise use the conditional common statistics.

Histograms Histograms can further improve the accuracy. Continuing the example above, we partition the values of the attribute A into buckets, and for each bucket store conditional common statistics: $\max_{a \in \texttt{Bucket}} ||\texttt{deg}_R(*|X, A = a)||_p$. There are 4 numbers per bucket. To support range predicates we need to store in each bucket the value $||\texttt{deg}_R(*|X, A \in \texttt{Bucket})||_p$.

Boolean Expressions The most powerful advantage of pessimistic estimators is that the Boolean connectives $\wedge, \vee$ simply becomes min and $+$. For example, if the query contains predicate $R.A = a \wedge R.B = b$, then we use as statistics $\min(||\texttt{deg}_R(*|X, A = a)||_p, ||\texttt{deg}_R(*|X, B = b)||_p)$; similarly, for $R.A = a \vee R.B = b$, we use[18] $+$. This principle extends to `IN` predicates (which are equivalent to $\bigvee$), and to `LIKE` predicates, which can be upper-bounded by a set of 3-grams, see [11].

Lossy Compression The complete degree sequence can be as large as the data, and therefore it is not usable as a statistic for cardinality estimation. Since DSB needs access to the entire degree sequence, it uses three observations to drastically reduce the size of the statistics. First, *run-length* compression is very effective on degree sequences: it compresses a sequence $\boldsymbol{a} = (a_1 \geq a_2 \geq \cdots)$ by replacing any constant subsequence $a_i = a_{i+1} = a_{i+2} = \cdots$ with the common value plus length. Second, DSB uses a lossy compression, replacing the original sequence $\boldsymbol{a}$ with $\boldsymbol{a}'$ such that $\boldsymbol{a} \leq \boldsymbol{a}'$ (element-wise) and the compression of $\boldsymbol{a}'$ is much smaller: since the DSB is monotone in the input degree sequences, it still returns an upper bound when $\boldsymbol{a}$ is replaced with $\boldsymbol{a}'$. However, since $||\boldsymbol{a}||_1 < ||\boldsymbol{a}'||_1$, the new sequence represents a relation with a larger cardinality. Instead, DSB upper bounds the CDF of the degree sequence, $A_i \stackrel{\text{def}}{=} \sum_{j \leq i} a_i$, instead of the PDF. That is, it uses a degree sequence $\boldsymbol{a}''$ such that $\boldsymbol{A} \leq \boldsymbol{A}''$ where $A''_i = \sum_{j \leq i} a''_i$. It is no longer obvious why DSB is still an upper bound when it uses $\boldsymbol{a}''$: we give the main intuition in Appendix C. Fig. 3 illustrates this method. The original sequence, $\boldsymbol{a} = (4, 2, 2, 1, 1, 1)$, can be compressed to $\boldsymbol{a}' = (4, 4, 4, 2, 2, 2)$, but the ℓ_1 norm increased from 11 to 18. Instead, DSB upper bounds the CDF to $\boldsymbol{A} \leq \boldsymbol{A}''$, see the second graph. The degree sequence that produces $\boldsymbol{A}''$ is $\boldsymbol{a}'' = (4, 3.5, 3.5, 0, 0, 0)$: notice that $\boldsymbol{a} \not\leq \boldsymbol{a}''$, yet by using $\boldsymbol{a}''$ DSB still returns an upper bound.

Runtime of the Pessimistic Estimator The runtime of DSB is linear in the size of the compressed representation of all degree sequences. The runtime of PolyB is exponential in n (the number of query variables, Eq. (6)), because the LP uses 2^n numerical variables. This is undesirable, but there are a few ways to mitigate this. First, when all statistics are for *simple* degree sequences (meaning of the form $\texttt{deg}_R(V|X)$, where X is a single variable, or $X = \emptyset$), then PolyB can be computed in PTIME, by using a totally different LP, with n^2 numerical variables instead of 2^n [21, 41]; furthermore, we only need $O(n)$ numerical variables in case of Berge acyclic queries [41]. Restricting the pessimistic estimator to simple degree sequences is quite reasonable, because many current systems already restrict their statistics to a single attribute, e.g. they store $|\mathsf{Dom}(R.X)|$, but rarely $|\mathsf{Dom}(R.XY)|$. When multi-attribute statistics are needed, then there are a few ways to optimize the LP that computes PolyB. First, ensure that it includes only the *elemental Shannon inequalities* [40, Chap.14.1]: there are $n(n-1)2^{n-3} + n$ elemental in-

[18] We need to require $p \geq 1$, because Minkowski's inequality $||\boldsymbol{a} + \boldsymbol{b}||_p \leq ||\boldsymbol{a}||_p + ||\boldsymbol{b}||_p$ only holds for $p \geq 1$.

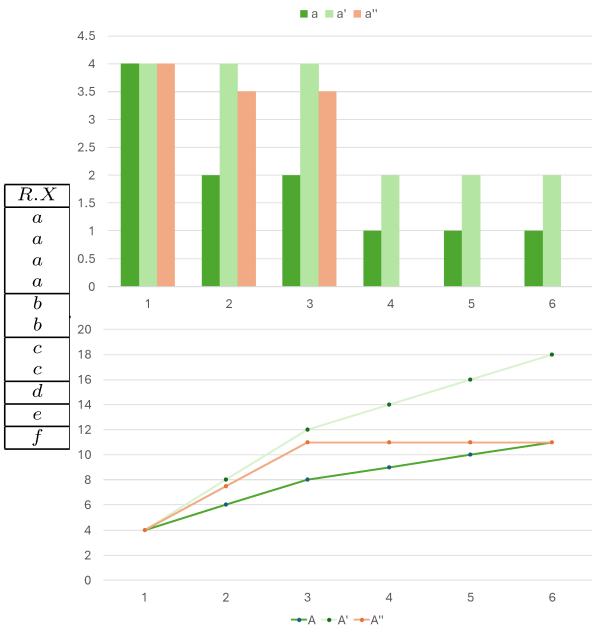


Figure 3: Illustration of the advanced compression in DSB: it shows the PDFs a, a', a'' and their CDFs $\boldsymbol{A}, \boldsymbol{A}', \boldsymbol{A}''$ (see text).

equalities. Second, drop unnecessary variables. For example, to compute the PolyB of a 2-way join $R(X,Y,Z,U,V) \bowtie S(V,W,K,L)$ it suffices to keep only one non-joining variable in each relation, i.e. simplify the query to $R(X,V) \bowtie S(V,W)$. Third, it is well known that creating the input for the LP solver often takes more time than the solver itself: to speedup the creation of the LP, the system can precompute and represent all elemental Shannon inequalities, since these depend only on n and not on the query.

11. A WISH COME TRUE?

We return now to our wish list in Sec. 3 and examine how pessimistic cardinality estimators do or do not satisfy them. The main advantage of PCE's is their *one-sided guarantee*. An application can count on the fact that the query output size will never exceed the PCE bound. The ability to provide a strong guarantee without any assumption is the main advantage of PCE over both traditional methods and ML-based methods. They are also *local*, in the sense that they only use statistics collected on each relation independently: if a new relation needs to be added to the database schema, then we only need to add statistics for that relation. This is similar to traditional CE's, and different from estimators based on trained models, which need access to the entire database.

The next question is how accurate the PCE's are. Only a few empirical evaluations have been done: for BoundSketch [4,5,34], DSB [11], and PolyB [41]. Out of the box, BoundSketch produces worse estimates than traditional estimators, however all implementations augment BoundSketch with hash-partitioning [4,34], which improves the accuracy, but at the cost of increased memory and significantly increased construction and estimation time.[19] While BoundSketch uses only cardinalities and max degrees, DSB and PolyB use the complete degree sequences (under lossy compression) and norms on these degree sequences, respectively, and produce much better bounds with less memory and with small estimation time [11, 41]. What several implementations report is that, when the traditional estimator in Postgre is replaced by BoundSketch, DSB, or PolyB, the performance of expensive SQL improves, sometimes significantly, because it causes the optimizer to be more cautions in choosing an execution plan. Cheap queries, however, tend to suffer from a regression.

In general, the empirical evaluation of PCE's is still very limited. Future research is needed to evaluate the impact of indices, on the essential query subexpressions, or of bitmap filtering, see [28] for an extensive discussion of the impact of the cardinality estimator on the query optimizer.

Moving down on our wish list, we note that DSB is *compositional*; in fact the algorithm computing DSB [10,11] is defined by recursion on the structure of a Berge-acyclic query. On the other hand, PolyB is *not* compositional. This appears to be unavoidable for cyclic queries, which do not admit a recursive definition. For example, computing the upper bounds on the join $R \bowtie S$ does not help us in deriving the bound $(|R| \cdot |S| \cdot |T|)^{1/2}$ of the 3-cycle C_3 (Eq. (18)). An open question is whether PolyB can be made compositional on (Berge-) acyclic queries.

Pessimistic cardinality estimators can be *combined* naturally. If we have two bounds on the query, B_1 and B_2, we can "combine" them by taking their min. Similar combinations can be done to handle complex predicates, consisting of `AND`, `OR`, `IN`, `LIKE`. We believe that this is the second main advantage of PCE's over traditional or ML-based methods. None of the other cardinality estimation methods have a natural way of combining estimates, and a recent study has found that different combination heuristics lead to quite different accuracies [5].

Finally, we consider the desire for *incremental updates*. Ideally, when the database is updated (via

[19] An improved partitioning is described recently in [9].

an insert/update/delete query), we would like to update the statistics incrementally. For context, we point out that, to the best of our knowledge, no relational database system updates its statistics incrementally, because it would slow down OLTP workloads, especially because updating a statistics requires acquiring a lock, which can quickly lead to high contention. A similar reason would prevent them to do incremental updates for pessimistic cardinality estimators. However, the question remains whether the data structures used by PCE's are able to support incremental updates. This appears to require materializing the degree sequence computed by Query (23), which is unlikely to be practical. Future work is needed to explore whether ℓ_p-sketches [2, 8] can be adapted for the purpose of incremental updates of the input statistics.

Acknowledgments We thank Nicolas Bruno, Surajit Chaudhuri, Bailu Ding, Benny Kimelfeld, Kukjin Lee, and Hung Ngo for their comments that helped improve the paper.

12. REFERENCES

[1] Statistics in SQL server. `https://learn.microsoft.com/en-us/sql/relational-databases/statistics/statistics?view=sql-server-ver16`, 2024. Accessed: October 2024.

[2] Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. In *STOC* (1996), pp. 20–29.

[3] Atserias, A., Grohe, M., and Marx, D. Size bounds and query plans for relational joins. *SIAM J. Comput. 42*, 4 (2013), 1737–1767.

[4] Cai, W., Balazinska, M., and Suciu, D. Pessimistic cardinality estimation: Tighter upper bounds for intermediate join cardinalities. In *SIGMOD* (2019), pp. 18–35.

[5] Chen, J., Huang, Y., Wang, M., Salihoglu, S., and Salem, K. Accurate summary-based cardinality estimation through the lens of cardinality estimation graphs. *Proc. VLDB Endow. 15*, 8 (2022), 1533–1545.

[6] Chen, Y., and Yi, K. Two-level sampling for join size estimation. In *SIGMOD* (2017), pp. 759–774.

[7] Cohen, E. Sampling big ideas in query optimization. In *PODS* (2023), pp. 361–371.

[8] Cormode, G., and Yi, K. *Small Summaries for Big Data*. Cambridge University Press, 2020.

[9] Deeds, K., Sabale, D., Kayali, M., and Suciu, D. Color: A framework for applying graph coloring to subgraph cardinality estimation. *CoRR abs/2405.06767* (2024).

[10] Deeds, K., Suciu, D., Balazinska, M., and Cai, W. Degree sequence bound for join cardinality estimation. In *ICDT* (2023), pp. 8:1–8:18.

[11] Deeds, K. B., Suciu, D., and Balazinska, M. Safebound: A practical system for generating cardinality bounds. *Proc. ACM Manag. Data 1*, 1 (2023), 53:1–53:26.

[12] Fagin, R. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM 30*, 3 (1983), 514–550.

[13] Friedgut, E., and Kahn, J. On the number of copies of one hypergraph in another. *Israel Journal of Mathematics 105*, 1 (1998), 251–256.

[14] Garcia-Molina, H., Ullman, J. D., and Widom, J. *Database systems - the complete book (2. ed.)*. Pearson Education, 2009.

[15] Gottlob, G., Lee, S. T., Valiant, G., and Valiant, P. Size and treewidth bounds for conjunctive queries. *J. ACM 59*, 3 (2012), 16:1–16:35.

[16] Grohe, M., and Marx, D. Constraint solving via fractional edge covers. In *SODA* (2006), pp. 289–298.

[17] Grohe, M., and Marx, D. Constraint solving via fractional edge covers. *ACM Trans. Alg. 11*, 1 (2014), 4:1–4:20.

[18] Han, Y., Wu, Z., Wu, P., Zhu, R., Yang, J., Tan, L. W., Zeng, K., Cong, G., Qin, Y., Pfadler, A., Qian, Z., Zhou, J., Li, J., and Cui, B. Cardinality estimation in DBMS: A comprehensive benchmark evaluation. *Proc. VLDB Endow. 15*, 4 (2021), 752–765.

[19] Hertzschuch, A., Hartmann, C., Habich, D., and Lehner, W. Simplicity done right for join ordering. In *CIDR* (2021).

[20] Im, S., Moseley, B., Ngo, H., and Pruhs, K. Tractable algorithms for join evaluation and cardinality estimation via the dual-project-dual technique, 2022. Personal communication.

[21] Im, S., Moseley, B., Ngo, H. Q., Pruhs, K., and Samadian, A. Optimizing polymatroid functions. *CoRR abs/2211.08381* (2022).

[22] Joglekar, M., and Ré, C. It's all a matter of degree: Using degree information to optimize multiway joins. In *ICDT* (2016), pp. 11:1–11:17.

[23] Khamis, M. A., Nakos, V., Olteanu, D., and Suciu, D. Join size bounds using l_p-norms on degree sequences. *CoRR abs/2306.14075* (2023).

[24] Khamis, M. A., Nakos, V., Olteanu, D., and Suciu, D. Join size bounds using l_p-norms on degree sequences. *Proc. ACM Manag. Data 2*, 2 (2024), 96.

[25] Khamis, M. A., Ngo, H. Q., and Suciu, D. Computing join queries with functional dependencies. In *PODS* (2016), pp. 327–342.

[26] Khamis, M. A., Ngo, H. Q., and Suciu, D. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? In *PODS* (2017), pp. 429–444. Extended version available at `http://arxiv.org/abs/1612.02503`.

[27] Kim, K., Jung, J., Seo, I., Han, W., Choi, K., and Chong, J. Learned cardinality estimation: An in-depth study. In *SIGMOD* (2022), pp. 1214–1227.

[28] Lee, K., Dutt, A., Narasayya, V. R., and Chaudhuri, S. Analyzing the impact of cardinality estimation on execution plans in microsoft SQL server. *Proc. VLDB Endow. 16*, 11 (2023), 2871–2883.

[29] Leis, V., Gubichev, A., Mirchev, A., Boncz, P. A., Kemper, A., and Neumann, T. How good are query optimizers, really? *Proc. VLDB Endow. 9*, 3 (2015), 204–215.

[30] Li, F., Wu, B., Yi, K., and Zhao, Z. Wander join: Online aggregation via random walks. In *SIGMOD* (2016), pp. 615–629.

[31] Li, F., Wu, B., Yi, K., and Zhao, Z. Wander join and XDB: online aggregation via random walks. *ACM Trans. Datab. Syst. 44*, 1 (2019), 2:1–2:41.

[32] Markl, V., Haas, P. J., Kutsch, M., Megiddo, N., Srivastava, U., and Tran, T. M. Consistent selectivity estimation via maximum entropy. *VLDB J. 16*, 1 (2007), 55–76.

[33] Ngo, H. Q. On an information theoretic approach to cardinality estimation (invited talk). In *ICDT* (2022), pp. 1:1–1:21.

[34] Park, Y., Ko, S., Bhowmick, S. S., Kim, K., Hong, K., and Han, W. G-CARE: A framework for performance benchmarking of cardinality estimation

techniques for subgraph matching. In *SIGMOD* (2020), pp. 1099–1114.
[35] Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. Access path selection in a relational database management system. In *SIGMOD* (1979), pp. 23–34.
[36] Suciu, D. Applications of information inequalities to database theory problems. In *LICS* (2023), pp. 1–30.
[37] Wang, X., Qu, C., Wu, W., Wang, J., and Zhou, Q. Are we ready for learned cardinality estimation? *Proc. VLDB Endow. 14*, 9 (2021), 1640–1654.
[38] Wu, Z., and Shaikhha, A. Bayescard: A unified bayesian framework for cardinality estimation. *CoRR abs/2012.14743* (2020).
[39] Yang, Z., Kamsetty, A., Luan, S., Liang, E., Duan, Y., Chen, X., and Stoica, I. Neurocard: One cardinality estimator for all tables. *Proc. VLDB Endow. 14*, 1 (2020), 61–73.
[40] Yeung, R. W. *Information Theory and Network Coding*, 1 ed. Springer Publishing Company, 2008.
[41] Zhang, H., Mayer, C., Khamis, M. A., Olteanu, D., and Suciu, D. LpBound: Pessimistic cardinality estimation using l_p-norms of degree sequences, 2024. technical report.
[42] Zhang, Z., and Yeung, R. W. On characterization of entropy function via information inequalities. *IEEE Trans. Inf. Theory 44*, 4 (1998), 1440–1452.

APPENDIX

A. PROOF OF THE CHAIN BOUND

We use the notations introduced in Sec. 7. Let $\boldsymbol{w}$ be a fractional cover, and let X_n be the last variable in the order π. We prove by induction on n:

$$E \stackrel{\text{def}}{=} \sum_{j=1,k} w_j h(V_j|U_j) \geq h(X_1 \cdots X_n) \qquad (26)$$

For $j \in J_1 \stackrel{\text{def}}{=} \{j \mid X_n \in V_j\}$, set $V_j' \stackrel{\text{def}}{=} V_j - \{X_n\}$; then $h(V_j|U_j) = h(V_j'|U_j) + h(X_n|U_jV_j')$. For $j \notin J_1$, let $V_j' \stackrel{\text{def}}{=} V_j$. Then $\boldsymbol{w}$ is a fractional cover of $(V_j'|U_j)$, and by induction we have $E \geq h(X_1 \cdots X_{n-1}) + \sum_{j\in J_1} w_j h(X_n|X_1 \cdots X_{n-1}) \geq h(X_1 \cdots X_n)$ because $\sum_{j\in J_1} w_j \geq 1$. The chain bound (10) follows from (26) using the standard argument from Sec. 8.

B. SOME INEQUALITY PROOFS

We prove (17)-(20). Inequality (17) follows from:

$$\begin{aligned}
&(p-2)\log|R| + 2\log||\mathtt{deg}_R(X|Y)||_2 \\
&\quad +(p-1)\log||\mathtt{deg}_S(Z|Y)||_{p-1} \\
&\quad +p\log||\mathtt{deg}_T(U|Z)||_p \\
&\geq (p-2)h(XY) + 2\left(\tfrac{1}{2}h(Y) + h(X|Y)\right) \\
&\quad +(p-1)\left(\tfrac{1}{p-1}h(Y) + h(Z|Y)\right) \\
&\quad +p\left(\tfrac{1}{p}h(Z) + h(U|Z)\right) \\
&= (p-1)h(XY) + h(X|Y) \\
&\quad +h(YZ) + (p-2)h(Z|Y) \\
&\quad +h(UZ) + (p-1)h(U|Z) \\
&= (p-2)\left[h(XY) + h(Z|Y) + h(U|Z)\right] \\
&\quad +\left[h(XY) + h(UZ)\right] \\
&\quad +\left[h(YZ) + h(X|Y) + h(U|Z)\right] \\
&\geq p \cdot h(XYZU) = p\log|J_3|
\end{aligned}$$

We have proven inequality (18) in Sec. 8. Inequalities (19) and (20) follows from:

$$\begin{aligned}
&(h(X) + 2h(Y|X)) + (h(Y) + 2h(Z|Y)) + (h(Z) + 2h(X|Z)) \\
&= (h(XY) + h(Y|X)) + (h(YZ) + h(Z|Y)) + (h(XZ) + h(X|Z)) \\
&\geq 3h(XYZ)
\end{aligned}$$

$$\begin{aligned}
&(h(X) + 3h(Y|X)) + (h(Z) + 3h(Y|Z)) + 5h(XZ) \\
&= (h(XY) + 2h(Y|X)) + (h(YZ) + 2h(Y|Z)) + 5h(XZ) \\
&= 2\left(h(XZ) + h(Y|X)\right) + 2\left(h(XZ) + h(Y|Z)\right) \\
&\qquad + (h(XY) + h(YZ) + h(XZ)) \\
&\geq 2h(XYZ) + 2h(XYZ) + 2h(XYZ) = 6h(XYZ)
\end{aligned}$$

C. CDF COMPRESSION

We prove here that the upper bound in Eq. (22) continues to hold if we replace the sequences $\boldsymbol{a}, \boldsymbol{b}$ with two sequences $\boldsymbol{a}', \boldsymbol{b}'$ whose CDFs are larger.

It will be convenient to use some notations. Given any sequence $\boldsymbol{x} = (x_1, x_2, \ldots)$, we denote by $\Delta\boldsymbol{x}$ and $\Sigma\boldsymbol{x}$ the following sequences:

$$(\Delta\boldsymbol{x})_i \stackrel{\text{def}}{=} x_i - x_{i-1} \qquad (\Sigma\boldsymbol{x})_i \stackrel{\text{def}}{=} \sum_{j=1,i} x_j$$

where $x_0 \stackrel{\text{def}}{=} 0$. If $\boldsymbol{x}$ is a probability distribution, meaning that $\sum_i x_i = 1$, then the sequence $\boldsymbol{x}$ is the *Probability Density Function* (PDF), and $\Sigma\boldsymbol{x}$ is the *Cumulative Density Function* (CDF). With some abuse, we use the terms PDF and CDF even when $\boldsymbol{x}$ is not a probability distribution. Obviously, $\Sigma\Delta\boldsymbol{x} = \Delta\Sigma\boldsymbol{x} = \boldsymbol{x}$.

The following *summation-by-parts* holds, for any two sequences $\boldsymbol{x}, \boldsymbol{y}$.

$$\sum_{i=1,n} (\Delta\boldsymbol{x})_i y_i = x_n y_n - \sum_{i=1,n-1} x_i (\Delta\boldsymbol{y})_{i+1}$$

Using this formula, we prove:

Lemma C.1. *Let $\boldsymbol{a}, \boldsymbol{b}$ be two, non-negative sequences, and assume that $\boldsymbol{b}$ is non-decreasing (meaning, $b_1 \geq b_2 \geq \cdots$). Let $\boldsymbol{A} \stackrel{def}{=} \Sigma\boldsymbol{a}$, and let $\boldsymbol{A}'$ be such that $\boldsymbol{A} \leq \boldsymbol{A}'$. Then, the following holds, where $\boldsymbol{a}' \stackrel{def}{=} \Delta\boldsymbol{A}'$:*

$$\sum_{i=1,n} a_i b_i \leq \sum_{i=1,n} a_i' b_i$$

The proof follows from:

$$\begin{aligned}
\sum_{i=1,n} a_i b_i &= \sum_{i=1,n} (\Delta\boldsymbol{A})_i b_i = A_n b_n - \sum_{i=1,n-1} A_i (\Delta\boldsymbol{b})_{i+1} \\
&\leq A_n' b_n - \sum_{i=1,n-1} A_i' (\Delta\boldsymbol{b})_{i+1}
\end{aligned}$$

which holds because $b_n \geq 0$ and $(\Delta\boldsymbol{b})_i \leq 0$. By applying the lemma a second time, we infer that the output of a join query (22) can be upper bounded by $\sum_i a_i' b_i'$, where $\boldsymbol{a}', \boldsymbol{b}'$ are the PDFs of $\boldsymbol{A}' \geq \boldsymbol{A}$ and $\boldsymbol{B}' \geq \boldsymbol{B}$.