Degree Sequence Bound for Join Cardinality Estimation

Kyle Deeds
University of Washington, Seattle, WA, USA

Dan Suciu
University of Washington, Seattle, WA, USA

Magda Balazinska
University of Washington, Seattle, WA, USA

Walter Cai
University of Washington, Seattle, WA, USA

Abstract

Recent work has demonstrated the catastrophic effects of poor cardinality estimates on query processing time. In particular, underestimating query cardinality can result in overly optimistic query plans which take orders of magnitude longer to complete than one generated with the true cardinality. Cardinality bounding avoids this pitfall by computing an upper bound on the query's output size using statistics about the database such as table sizes and degrees, i.e. value frequencies. In this paper, we extend this line of work by proving a novel bound called the Degree Sequence Bound which takes into account the full degree sequences and the max tuple multiplicity. This work focuses on the important class of Berge-Acyclic queries for which the Degree Sequence Bound is tight. Further, we describe how to practically compute this bound using a functional approximation of the true degree sequences and prove that even this functional form improves upon previous bounds.

2012 ACM Subject Classification Information systems → Query optimization; Information systems → Query planning; Theory of computation → Database query processing and optimization (theory); Theory of computation → Data modeling

Keywords and phrases Cardinality Estimation, Cardinality Bounding, Degree Bounds, Functional Approximation, Query Planning, Berge-Acyclic Queries

Digital Object Identifier 10.4230/LIPIcs.ICDT.2023.8

Related Version *Full Version*: `https://arxiv.org/pdf/2201.04166` [4]

Funding This work is supported by National Science Foundation grants NSF IIS 1907997 and NSF-BSF 2109922.

1 Introduction

The weakest link in a modern query processing engine is the *cardinality estimator*. There are several major decisions where the system needs to estimate the size of a query's output: the optimizer uses the estimate to compute an effective query plan; the scheduler needs the estimate to determine how much memory to allocate for a hash table and to decide whether to use a main-memory or an out-of-core algorithm; a distributed system needs the estimate to decide how many servers to reserve for subsequent operations. Today's systems estimate the cardinality of a query by making several strong and unrealistic assumptions, such as uniformity and independence. As a result, the estimates for multi-join queries commonly have relative errors up to several orders of magnitude. An aggravating phenomenon is that cardinality estimators consistently underestimate (this is a consequence of the independence assumption), and this leads to wrong decisions for the most expensive queries [15, 3, 10]. A significant amount of effort has been invested in the last few years into using machine

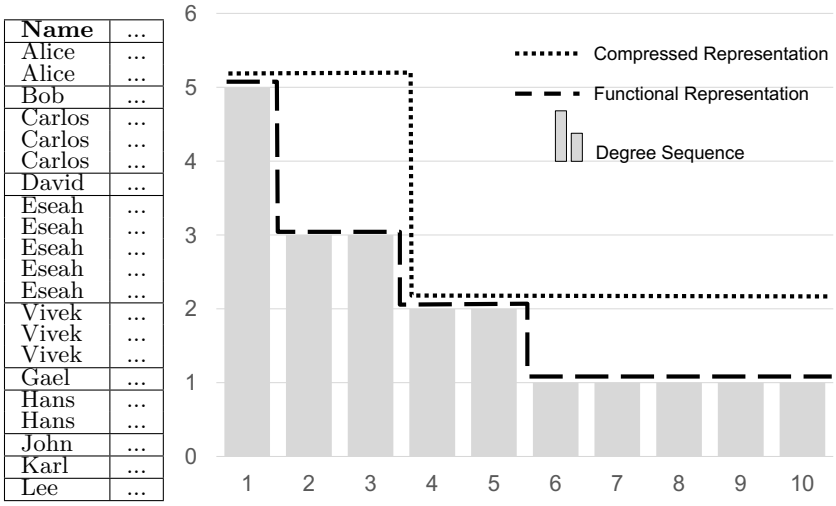

Figure 1 The degree sequence of `Name`. The first rank represents `Eseah` whose degree is 5, the next two ranks are for `Carlos` and `Vivek` whose degrees are 3. The degree sequence can be represented compactly using a staircase functions, and even more compactly using lossy compression.

learning for cardinality estimation [20, 22, 23, 24, 21, 16, 17], but this approach still faces several formidable challenges, such as the need for large training sets, the long training time of complex models, and the lack of guarantees about the resulting estimates.

An alternative approach to estimating the cardinality is to compute an *upper bound* for the size of the query answer. This approach originated in the database theory community, through the pioneering results by Grohe and Marx [8] and Atserias, Grohe, and Marx [1]. They described an elegant formula, now called the AGM bound, that gives a tight upper bound on the query result in terms of the cardinalities of the input tables. This upper bound was improved by the *polymatroid bound*, which takes into account both the cardinalities, and the degree constraints and includes functional dependencies as a special case [7, 13, 14, 18]. In principle, an upper bound could be used by a query optimizer in lieu of a cardinality estimator and, indeed, this idea was recently pursued by the systems community, where the upper bound appears under various names such as bound sketch or pessimistic cardinality estimator [3, 11]. In this paper, we will call it a *cardinality bound*. As expected, a cardinality bound prevents query optimizers from choosing disastrous plans for the most expensive queries [3], however, their relative error is often much larger than that of other methods [19, 6]. While the appeal of a guaranteed upper bound is undeniable, in practice overly pessimistic bounds are unacceptable.

In this paper, we propose a new upper bound on the query size based on *degree sequences*. By using a slightly larger memory footprint, this method has the potential to achieve much higher accuracy than previous bounds. Given a relation R, an attribute X, and a value $u \in \Pi_X(R)$, the *degree* of u is the number of tuples in R with u in the X attribute, formally $d^{(u)} = |\sigma_{X=u}(R)|$. The *degree sequence* of an attribute X in relation R is the sorted sequence of all degrees for the values of that attribute, $d^{(u_1)} \geq d^{(u_2)} \geq \cdots \geq d^{(u_n)}$. Going forward, we drop any reference to values and instead refer to degrees by their index in this sequence, also called their *rank*, i.e. $d_1 \geq \cdots \geq d_n$.[1] A degree sequence can easily be computed

[1] Note that the degree sequence is very similar to a rank-frequency distribution in the probability literature and has been extensively used in graph analysis [2, 9].

offline, and can be compressed effectively, with a good space/accuracy tradeoff due to its monotonicity; see Fig. 1 for an illustration. Degree sequences offer more information on the database instance than the statistics used by previous upper bounds. For example, the AGM bound uses only the cardinality of the relations, which is $\sum_i d_i$, while the extension to degree constraints [14] uses the cardinality, $\sum_i d_i$, and the maximum degree, d_1.

For this new bound we had to develop entirely new techniques over those used for the AGM and the polymatroid bounds. Previous techniques are based on information theory. If some relation $R(X, Y)$ has cardinality N, then any probability space over R has an entropy that satisfies $H(XY) \leq \log N$; if the degree sequence of the attribute X is $d_1 \geq d_2 \geq \ldots$, then $H(Y|X) \leq \log d_1$. Both the AGM and the polymatroid bound start from such constraints on the entropy. Unfortunately, these constraints do not extend to degree sequences, because H is ignorant of $d_2, d_3, \ldots$ Information theory gives us only three degrees of freedom, namely $H(XY), H(X), H(Y)$, while the degree sequence has an arbitrary number of degrees of freedom. Rather than using information theory, our new framework models relations as tensors, and formulates the upper bound as a linear optimization problem. This framework is restricted to *Berge-acyclic, fully conjunctive queries* [5] (reviewed in Sec. 2); throughout the paper we will assume that queries are in this class. As we explain in Appendix A.1 [4] these are the most common queries found in applications.

The Worst-Case Instance. Our main result (Theorems 3.2 and 4.1) is a tight cardinality bound given the degree sequences of all relations. This bound is obtained by evaluating the query on a *worst-case instance* that satisfies those degree constraints.[2] Intuitively, each relation of the worst-case instance is obtained by matching the highest degree values in the different columns, and the same principle is applied across relations. For example, consider the join $R(X, \ldots) \bowtie S(X, \ldots)$, where the degree sequences of $R.X$ and $S.X$ are $a_1 \geq a_2 \geq \cdots$ and $b_1 \geq b_2 \geq \cdots$ respectively. The true cardinality of the join is $\sum_i a_i b_{\tau(i)}$ for some unknown permutation τ , while the maximum cardinality is[3] $\sum_i a_i b_i$, and is obtained when the highest degree values match. Our degree sequence bound holds even when the input relations are allowed to be bags. Furthermore, we prove (Theorem 4.6) that this bound is always below the AGM and polymatroid bounds, although the latter restrict the relations to be sets. To prove this we had to develop a new, explicit formula for the polymatroid bound for Berge-acyclic queries, which is of independent interest (Theorem 4.3).

Compact Representation. A full degree sequence is about as large as the relation instance, while cardinality estimators need to run in sub-linear time. Fortunately, a degree sequence can be represented compactly using a piece-wise constant function, called a *staircase function*, as illustrated in Fig. 1. Our next result, Theorem 5.2, is an algorithm for the degree sequence bound that runs in quasi-linear time (i.e. linear plus a logarithmic factor) in the size of the representation, independent of the size of the instance. The algorithm makes some rounding errors (Lemma 5.1), hence its output may be slightly larger than the exact bound, however we prove that it is still lower than the AGM and polymatroid bounds (Theorem 5.5). The algorithm can be used in conjunction with a compressed representation of the degree sequence. By using few buckets and upper-bounding the degree sequence one can trade off the memory size and estimation time for accuracy. At one extreme, we could upper bound

[2] In graph theory, the problem of computing a graph satisfying a given degree sequence is called the *realization problem*.

[3] For example, if $a_1 \geq a_2$, $b_1 \geq b_2$, then $a_1b_1 + a_2b_2 \geq a_1b_2 + a_2b_1$.

the entire sequence using a single bucket with the constant d_1, at the other extreme we could keep the complete sequence. Neither the AGM bound nor the polymatroid bound have this tradeoff ability.

Max Tuple Multiplicity. Despite using more information than previous upper bounds, our bound can still be overly pessimistic, because it needs to match the most frequent elements in all attributes. For example, suppose a relation has two attributes whose highest degrees are a_1 and b_1 respectively. Its worst-case instance is a bag and must include some tuple that occurs $\min(a_1, b_1)$ times. Usually, a_1 and b_1 are large, since they represent the frequencies of the worst heavy hitters in the two columns, but in practice they rarely occur together $\min(a_1, b_1)$ times. To avoid such worst-case matchings, we use one additional piece of information on each base table: the max multiplicity over all tuples, denoted B. Usually, B is significantly smaller than the largest degrees, and, by imposing it as an additional constraint, we can significantly improve the query's upper bound; in particular, when $B = 1$ then the relation is restricted to be a set. Our main results in Theorems 3.2 and 4.1 extend to max tuple multiplicities, but in some unexpected ways. The worst-case relation, while still *tight*, is not a conventional relation: it may have tuples that occur more than B times, and, when the relation has 3 or more attributes it may even have tuples with negative multiplicities. Nevertheless, these rather unconventional worst-case relations provide an even better degree sequence bound than by ignoring B.

▶ **Example 1.1.** To give a taste of our degree-sequence bound, consider the full conjunctive query $Q(\cdots) = R(X, \cdots) \bowtie S(X, Y, \cdots) \bowtie T(Y, \cdots)$, where we omit showing attributes that appear in only one of the relations. Alternatively, we can write $Q(X, Y) = R(X) \bowtie S(X, Y) \bowtie T(Y)$ where R, S, T are bags rather than sets. Assume the following degree sequences:

$$\boldsymbol{d}^{(R.X)} = (3, 2, 2) \qquad \boldsymbol{d}^{(T.Y)} = (2, 1, 1, 1) \qquad \boldsymbol{d}^{(S.X)} = (5, 1) \qquad \boldsymbol{d}^{(S.Y)} = (3, 2, 1) \tag{1}$$

The AGM bound uses only the cardinalities, which are:

$$|R| = 7 \qquad |S| = 6 \qquad |T| = 5$$

The AGM bound[4] is $|R| \cdot |S| \cdot |T| = 210$. The extension to degree constraints in [14] uses in addition the maximum degrees:

$$\texttt{deg}(R.X) = 3 \qquad \texttt{deg}(S.X) = 5 \qquad \texttt{deg}(S.Y) = 3 \qquad \texttt{deg}(T.Y) = 2$$

and the bound is the minimum between the AGM bound and the following quantities:

$$|R| \cdot \texttt{deg}(S.X) \cdot \texttt{deg}(T.Y) = 7 \cdot 5 \cdot 2 = 70$$
$$\texttt{deg}(R.X) \cdot |S| \cdot \texttt{deg}(T.Y) = 3 \cdot 6 \cdot 2 = 36$$
$$\texttt{deg}(R.X) \cdot \texttt{deg}(S.Y) \cdot |T| = 3 \cdot 3 \cdot 5 = 45$$

Thus, the degree-constraint bound is improved to 36.

Our new bound is given by the answer to the query on the worst-case instance of the relations R, S, T, shown here together with their multiplicities (recall that they are bags):

$$R = \begin{array}{|c|}\hline a \\ \hline b \\ \hline c \\ \hline \end{array} \begin{array}{c} 3 \\ 2 \\ 2 \end{array}, \qquad S = \begin{array}{|c|c|}\hline a & u \\ \hline a & v \\ \hline b & w \\ \hline \end{array} \begin{array}{c} 3 \\ 2 \\ 1 \end{array}, \qquad T = \begin{array}{|c|}\hline u \\ \hline v \\ \hline w \\ \hline z \\ \hline \end{array} \begin{array}{c} 2 \\ 1 \\ 1 \\ 1 \end{array},$$

[4] Recall that each of the three relations has private variables, e.g. $R(X, U), S(X, Y, V, W), T(Y, Z)$. The only fractional edge cover is $1, 1, 1$.

The three relations have the required degree sequences, for example $S.X$ consists of 5 a's and 1 b, thus has degree sequence $(5, 1)$. Notice the matching principle: we assumed that the most frequent element in $R.X$ and $S.X$ are the same value a, and that the most frequent values in $S.X$ and in $S.Y$ occur together. On this instance, we compute the query and obtain the answer Q.

$$Q = \begin{array}{|c|c|l} \hline a & u & 3 \cdot 3 \cdot 2 = 18 \\ a & v & 3 \cdot 2 \cdot 1 = 6 \\ b & w & 2 \cdot 1 \cdot 1 = 2 \\ \hline \end{array} \qquad S' = \begin{array}{|c|c|l} \hline a & u & 2 \\ a & v & 2 \\ a & w & 1 \\ b & u & 1 \\ \hline \end{array}$$

The upper bound is the size of the answer on this instance, which is $18 + 6 + 2 = 26$, and it improves over 36. Here, the improvement is relatively minor, but this is a consequence of the short example. In practice, degree sequences often have a long tail, i.e. with a few large leading degrees $d_1, d_2, \ldots$ followed by very many small degrees $d_m, d_{m+1}, \ldots, d_n$ (with a large n). In that case the improvements of the new bound can be very significant.

Suppose now that we have one additional information about S: every tuple occurs at most $B = 2$ times. Then we need to reduce the multiplicity of (a, u), and the new worst-case instance, denoted S', is the following relation which decreases the cardinality bound to 25.

2 Problem Statement

Tensors. In this paper, it is convenient to define tensors using a named perspective, where each dimension is associated with a variable. We write variables with capital letters $X, Y, \ldots$ and sets of variables with boldface, $\boldsymbol{X}, \boldsymbol{Y}, \ldots$ We assume that each variable X has an associated finite domain $D_X \stackrel{\text{def}}{=} [n_X]$ for some number $n_X \geq 1$. For any set of variables $\boldsymbol{X}$ we denote by $D_{\boldsymbol{X}} \stackrel{\text{def}}{=} \prod_{Z \in \boldsymbol{X}} D_Z$. We use lower case for values, e.g. $z \in D_Z$ and boldface for tuples, e.g. $\boldsymbol{x} \in D_{\boldsymbol{X}}$. An *$\boldsymbol{X}$-tensor*, or simply a tensor when $\boldsymbol{X}$ is clear from the context, is $\boldsymbol{M} \in \mathbb{R}^{D_{\boldsymbol{X}}}$. We say that $\boldsymbol{M}$ has $|\boldsymbol{X}|$ dimensions. Given two $\boldsymbol{X}$-tensors $\boldsymbol{M}, \boldsymbol{N}$, we write $\boldsymbol{M} \leq \boldsymbol{N}$ for the component-wise order ($M_{\boldsymbol{x}} \leq N_{\boldsymbol{x}}$, for all $\boldsymbol{x}$). If $\boldsymbol{X}, \boldsymbol{Y}$ are two sets of variables, then we denote their union by $\boldsymbol{XY}$. If, furthermore, $\boldsymbol{X}, \boldsymbol{Y}$ are disjoint, and $\boldsymbol{x} \in D_{\boldsymbol{X}}, \boldsymbol{y} \in D_{\boldsymbol{Y}}$, then we denote by $\boldsymbol{xy} \in D_{\boldsymbol{XY}}$ the concatenation of the two tuples.

▶ **Definition 2.1.** Let $\boldsymbol{M}, \boldsymbol{N}$ be an $\boldsymbol{X}$-tensor, and a $\boldsymbol{Y}$-tensor respectively. Their tensor product is the following $\boldsymbol{XY}$-tensor:

$$\forall \boldsymbol{z} \in D_{\boldsymbol{XY}} : \qquad (\boldsymbol{M} \otimes \boldsymbol{N})_{\boldsymbol{z}} \stackrel{\text{def}}{=} M_{\pi_{\boldsymbol{X}}(\boldsymbol{z})} \cdot N_{\pi_{\boldsymbol{Y}}(\boldsymbol{z})} \tag{2}$$

If $\boldsymbol{X}, \boldsymbol{Y}$ are disjoint and $\boldsymbol{M}$ is an $\boldsymbol{XY}$-tensor then we define its $\boldsymbol{X}$-summation to be the following $\boldsymbol{Y}$-tensor:

$$\forall \boldsymbol{y} \in D_{\boldsymbol{Y}} : \qquad (\texttt{SUM}_{\boldsymbol{X}}(\boldsymbol{M}))_{\boldsymbol{y}} \stackrel{\text{def}}{=} \sum_{\boldsymbol{x} \in D_{\boldsymbol{X}}} M_{\boldsymbol{xy}} \tag{3}$$

If $\boldsymbol{M}, \boldsymbol{N}$ are $\boldsymbol{XY}$ and $\boldsymbol{YZ}$ tensors, where $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ are disjoint sets of variables, then their dot product is the $\boldsymbol{XZ}$-tensor:

$$\forall \boldsymbol{x} \in D_{\boldsymbol{X}}, \boldsymbol{z} \in D_{\boldsymbol{Z}} : \qquad (\boldsymbol{M} \cdot \boldsymbol{N})_{\boldsymbol{xz}} \stackrel{\text{def}}{=} \texttt{SUM}_{\boldsymbol{Y}}(\boldsymbol{M} \otimes \boldsymbol{N})_{\boldsymbol{xz}} = \sum_{\boldsymbol{y} \in D_{\boldsymbol{Y}}} M_{\boldsymbol{xy}} N_{\boldsymbol{yz}} \tag{4}$$

In other words, in this paper we use $\otimes$ like a natural join. For example, if $\boldsymbol{M}$ is an IJ-tensor (i.e. a matrix) and $\boldsymbol{N}$ is an KL-tensor, then $\boldsymbol{M} \otimes \boldsymbol{N}$ is the Kronecker product; if $\boldsymbol{P}$ is an IJ-tensor (like $\boldsymbol{M}$) then $\boldsymbol{M} \otimes \boldsymbol{P}$ is the element-wise product. The dot product

sums out the common variables, for example if $\boldsymbol{a}$ is a J-tensor, then $\boldsymbol{M} \cdot \boldsymbol{a}$ is the standard matrix-vector multiplication, and its result is an I-tensor. The following is easily verified. If $\boldsymbol{M}$ is an $\boldsymbol{X}$-tensor, $\boldsymbol{N}$ is a $\boldsymbol{Y}$-tensor and $\boldsymbol{X}, \boldsymbol{Y}$ are disjoint sets of variables, then:

$$\forall \boldsymbol{X}_0 \subseteq \boldsymbol{X}, \forall \boldsymbol{Y}_0 \subseteq \boldsymbol{Y}: \qquad \texttt{SUM}_{\boldsymbol{X}_0\boldsymbol{Y}_0}(\boldsymbol{M} \otimes \boldsymbol{N}) = \texttt{SUM}_{\boldsymbol{X}_0}(\boldsymbol{M}) \otimes \texttt{SUM}_{\boldsymbol{Y}_0}(\boldsymbol{N}) \tag{5}$$

Permutations. A permutation on $D = [n]$ is a bijective function $\sigma : D \to D$; the set of permutations on D is denoted S_D, or simply S_n. If $\boldsymbol{D} = D_1 \times \cdots \times D_k$ then we denote by $S_{\boldsymbol{D}} \stackrel{\text{def}}{=} S_{D_1} \times \cdots \times S_{D_k}$. Given an $\boldsymbol{X}$-tensor $\boldsymbol{M} \in \mathbb{R}^{D_{\boldsymbol{X}}}$ and permutations $\boldsymbol{\sigma} \in S_{D_{\boldsymbol{X}}}$, the *$\boldsymbol{\sigma}$-permuted $\boldsymbol{X}$-tensor* is $\boldsymbol{M} \circ \boldsymbol{\sigma} \in \mathbb{R}^{D_{\boldsymbol{X}}}$:

$$\forall \boldsymbol{x} \in D_{\boldsymbol{X}}: \qquad (\boldsymbol{M} \circ \boldsymbol{\sigma})_{\boldsymbol{x}} \stackrel{\text{def}}{=} \boldsymbol{M}_{\boldsymbol{\sigma}(\boldsymbol{x})}$$

Sums are invariant under permutations, for example if $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^{D_Z}$ are Z-vectors and $\sigma \in S_{D_Z}$, then $(\boldsymbol{a} \circ \sigma) \cdot (\boldsymbol{b} \circ \sigma) = \boldsymbol{a} \cdot \boldsymbol{b}$, because $\sum_{i \in D_Z} a_{\sigma(i)} b_{\sigma(i)} = \sum_{i \in D_Z} a_i b_i$.

Queries. A full conjunctive query Q is:

$$Q(\boldsymbol{X}) = \bowtie_{R \in \boldsymbol{R}} R(\boldsymbol{X}_R) \tag{6}$$

where $\boldsymbol{R} \stackrel{\text{def}}{=} \boldsymbol{R}(Q)$ denotes the set of its relations, $\boldsymbol{X}$ is a set of variables, and $\boldsymbol{X}_R \subseteq \boldsymbol{X}$ for each relation $R \in \boldsymbol{R}$. The *incidence graph* of Q is the following bipartite graph: $T \stackrel{\text{def}}{=} (\boldsymbol{R} \cup \boldsymbol{X}, E \stackrel{\text{def}}{=} \{(R, Z) \mid Z \in \boldsymbol{X}_R\})$. It can be shown that Q is *Berge-acyclic* [5] iff its incidence graph is an undirected tree (see Appendix A.1 [4]). Unless otherwise stated, all queries in this paper are assumed to be full, Berge-acyclic conjunctive queries. We use bag semantics for query evaluation, and represent an *instance* of a relation $R \in \boldsymbol{R}$ by an $\boldsymbol{X}_R$-tensor, $\boldsymbol{M}^{(R)}$, where $M^{(R)}_t$ is defined to be the multiplicity of the tuple $t \in D_{\boldsymbol{X}_R}$ in the bag R. The number of tuples in the answer to Q is:

$$|Q| = \texttt{SUM}_{\boldsymbol{X}} \left(\bigotimes_{R \in \boldsymbol{R}} \boldsymbol{M}^{(R)} \right) \tag{7}$$

▶ **Example 2.2.** Consider the following query:

$$Q(X, Y, Z, U, V, W) = R(X, Y) \bowtie S(Y, Z, U) \bowtie T(U, V) \bowtie K(Y, W)$$

Its incidence graph is $T = (\{R, \ldots, K\} \cup \{X, \ldots, W\}, \{(R, X), (R, Y), (S, Y), \ldots, (K, W)\})$ and is an undirected tree. An instance of $R(X, Y)$ is represented by a matrix $\boldsymbol{M}^{(R)} \in \mathbb{R}^{D_X \times D_Y}$, where $M^{(R)}_{xy}$ = the number of times the tuple (x, y) occurs in R. Similarly, S is represented by a tensor $\boldsymbol{M}^{(S)} \in \mathbb{R}^{D_Y \times D_Z \times D_U}$. The size of the query's output is:

$$|Q| = \sum_{x,y,z,u,v,w} M^{(R)}_{xy} M^{(S)}_{yzu} M^{(T)}_{uv} M^{(K)}_{yw}$$

Degree Sequences. We denote by $\mathbb{R}_+ \stackrel{\text{def}}{=} \{x \mid x \in \mathbb{R}, x \geq 0\}$ and we say that a vector $f \in \mathbb{R}_+^{[n]}$ is non-increasing if $f_{r-1} \geq f_r$ for $r \in [2, \ldots, n]$.

▶ **Definition 2.3.** Fix a set of variables $\boldsymbol{X}$, with domains D_Z, $Z \in \boldsymbol{X}$. A *degree sequence* associated with the dimension $Z \in \boldsymbol{X}$ is a non-increasing vector $\boldsymbol{f}^{(Z)} \in \mathbb{R}_+^{D_Z}$. We call the index r the *rank*, and $f^{(Z)}_r$ the *degree at rank r*. An $\boldsymbol{X}$-tensor $\boldsymbol{M}$ is *consistent* w.r.t. $\boldsymbol{f}^{(Z)}$ if:

$$\texttt{SUM}_{\boldsymbol{X}-\{Z\}}(\boldsymbol{M}) \leq \boldsymbol{f}^{(Z)} \tag{8}$$

$\boldsymbol{M}$ is consistent with a tuple of degree sequences $\boldsymbol{f}^{(\boldsymbol{X})} \stackrel{\text{def}}{=} (\boldsymbol{f}^{(Z)})_{Z \in \boldsymbol{X}}$, if it is consistent with every $\boldsymbol{f}^{(Z)}$. Furthermore, given $B \in \mathbb{R}_+ \cup \{\infty\}$, called the *max tuple multiplicity*, we say that $\boldsymbol{M}$ is *consistent* w.r.t. B if $\boldsymbol{M}_t \leq B$ for all $t \in D_{\boldsymbol{X}}$. We denote:

$$\begin{aligned}\mathcal{M}_{\boldsymbol{f}^{(\boldsymbol{X})},B} &\stackrel{\text{def}}{=} \{\boldsymbol{M} \in \mathbb{R}^{D_{\boldsymbol{X}}} \mid \boldsymbol{M} \text{ is consistent with } \boldsymbol{f}^{(\boldsymbol{X})}, B\} \\ \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})},B} &\stackrel{\text{def}}{=} \{\boldsymbol{M} \in \mathbb{R}_+^{D_{\boldsymbol{X}}} \mid \boldsymbol{M} \text{ is non-negative and consistent with } \boldsymbol{f}^{(\boldsymbol{X})}, B\}\end{aligned} \tag{9}$$

For a simple illustration consider two degree sequences $\boldsymbol{f} \in \mathbb{R}^{[m]}, \boldsymbol{g} \in \mathbb{R}^{[n]}$. $\mathcal{M}_{\mathbf{f},\mathbf{g},\infty}$ is the set of matrices $\boldsymbol{M}$ whose row-sums and column-sums are $\leq \boldsymbol{f}$ and $\leq \boldsymbol{g}$ respectively; $\mathcal{M}^+_{\mathbf{f},\mathbf{g},\infty}$ is the subset of non-negative matrices; $\mathcal{M}^+_{\mathbf{f},\mathbf{g},B}$ is the subset of matrices that also satisfy $M_{ij} \leq B$, $\forall i, j$.

Problem Statement. Fix a query Q. For each relation R, we are given a set of degree sequences $\boldsymbol{f}^{(R,\boldsymbol{X}_R)} \stackrel{\text{def}}{=} \left(\boldsymbol{f}^{(R,Z)}\right)_{Z \in \boldsymbol{X}_R}$, and a tuple multiplicity $B^{(R)} \in \mathbb{R}_+ \cup \{\infty\}$. We are asked to find the maximum size of Q over all database instances consistent with all degree sequences and tuple multiplicities. To do this, we represent a relation instance R by an unknown tensor $\boldsymbol{M}^{(R)} \in \mathcal{M}^+_{\mathbf{f}^{(R,\boldsymbol{X}_R)},B^{(R)}}$ and an unknown set of permutations $\boldsymbol{\sigma}^{(R)} \in S_{D_{\boldsymbol{X}_R}}$, and solve the following problem:

▶ **Problem 1** (Degree Sequence Bound)**.** Solve the following optimization problem:

$$\begin{aligned}&\text{Maximize: } |Q| = \mathtt{SUM}_{\boldsymbol{X}}\left(\bigotimes_{R \in \boldsymbol{R}} (\boldsymbol{M}^{(R)} \circ \boldsymbol{\sigma}^{(R)})\right) \\ &\text{Where: } \forall R \in \boldsymbol{R},\ \boldsymbol{\sigma}^{(R)} \in S_{D_{\boldsymbol{X}_R}},\ \boldsymbol{M}^{(R)} \in \mathcal{M}^+_{\boldsymbol{f}^{(R,\boldsymbol{X}_R)},B^{(R)}}\end{aligned} \tag{10}$$

This is a non-linear optimization problem: while the set $\mathcal{M}^+$ defined in Eq. (9) is a set of linear constraints, the objective (10) is non-linear. In the rest of the paper we describe an explicit formula for the degree sequence bound, which is optimal (i.e. tight) when $B^{(R)} = \infty$, for all R, and is optimal in a weaker sense in general.

▶ **Example 2.4.** Continuing Example 1.1, the four degree sequences in (1) correspond to the variables in each relation $R.X$, $S.X$, $S.Y$, and $T.Y$. Since $S.X$ has a shorter degree sequence than $R.X$, we pad it with a 0, so it becomes $\boldsymbol{d}^{(S.X)} = (5, 1, 0)$; similarly for $\boldsymbol{d}^{(S.Y)}$. Instead of values c, b, a, we use indices $1, 2, 3$, similarly u, v, w, z becomes $1, 2, 3, 4$. For example, $S = \begin{array}{|c|c|l}\hline 3 & 1 & 3 \\ 3 & 2 & 2 \\ 2 & 3 & 1 \\ \hline\end{array}$ is isomorphic to the instance in Example 1.1. It is represented by $\boldsymbol{M} \circ (\sigma, \tau)$ where the matrix $\boldsymbol{M} = \begin{pmatrix}3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0\end{pmatrix}$, (its row-sums are $5, 1, 0$ and column-sums are $3, 2, 1, 0$, as required) and the permutations are, in two-line notation, $\sigma \stackrel{\text{def}}{=} \begin{pmatrix}1 & 2 & 3 \\ 3 & 2 & 1\end{pmatrix}$ and $\tau \stackrel{\text{def}}{=}$ the identity. Similarly, the relations R, T, are represented by vectors $\boldsymbol{a}, \boldsymbol{b}$ and permutations θ, ρ. The bound of Q is the maximum value of $\sum_{i=1,3} \sum_{j=1,4} M_{\sigma(i)\tau(j)} a_{\theta(i)} b_{\rho(j)}$, where $\boldsymbol{M}, \boldsymbol{a}, \boldsymbol{b}$ are consistent with the given degree sequences, and $\sigma, \tau, \theta, \rho$ are permutations. This is a special case of Eq. (10).

3 The Star Query

We start by computing the degree sequence bound for a *star query*, which is defined as:

$$Q_{\texttt{star}} = S(X_1, \ldots, X_d) \bowtie R^{(1)}(X_1) \bowtie \cdots \bowtie R^{(d)}(X_d) \tag{11}$$

Assume that the domain of each variable X_p is $[n_p]$ for some $n_p > 0$, and denote by $[\boldsymbol{n}] \stackrel{\text{def}}{=} [n_1] \times \cdots \times [n_d]$. Later, in Sec. 4, we will use the bound for $Q_{\texttt{star}}$ as a building block to compute the degree sequence bound of a general query Q. There, S will be one of the relations of the query, for which we know the degree sequences $\boldsymbol{f}^{(X_p)} \in \mathbb{R}_+^{[n_p]}$, $p = 1, \ldots, d$ and tuple bound B, while the unary relations $R^{(1)}, \ldots, R^{(d)}$ will be results of subqueries, which are unknown. The instance of each $R^{(p)}$ is given by an unknown vector $\boldsymbol{a}^{(p)} \in \mathbb{R}_+^{[n_p]}$, which we can assume w.l.o.g. to be non-increasing, by permuting the domain of X_p in both S and in $R^{(p)}$. Therefore, S will be represented by $\boldsymbol{M} \circ \boldsymbol{\sigma}$, where $\boldsymbol{M} \in \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})}, B}$ is some tensor and $\boldsymbol{\sigma}$ some permutation, and the size of $Q_{\texttt{star}}$ is:

$$|Q_{\texttt{star}}| = \sum_{(i_1,\ldots,i_d) \in [\boldsymbol{n}]} (\boldsymbol{M} \circ \boldsymbol{\sigma})_{i_1 \cdots i_d} \cdot a_{i_1}^{(X_1)} \cdots a_{i_d}^{(X_d)} \tag{12}$$

Equivalently: $|Q_{\texttt{star}}| = \texttt{SUM}_{\boldsymbol{X}} \left((\boldsymbol{M} \circ \boldsymbol{\sigma}) \otimes \bigotimes_p \boldsymbol{a}^{(X_p)} \right) = (\boldsymbol{M} \circ \boldsymbol{\sigma}) \cdot \boldsymbol{a}^{(X_1)} \cdots \boldsymbol{a}^{(X_d)}$.

Our goal is to find the unknown $\boldsymbol{M} \circ \boldsymbol{\sigma}$ for which $|Q_{\texttt{star}}|$ is maximized, no matter what the unary relations are. It turns out that $\boldsymbol{\sigma}$ can always be chosen the identity permutation, thus it remains to find the optimal $\boldsymbol{M}$, which we denote by $\boldsymbol{C}$. This justifies:

▶ **Problem 2** (Worst-Case Tensor)**.** Fix $\boldsymbol{f}^{(\boldsymbol{X})}$, B. Find a tensor $\boldsymbol{C} \in \mathcal{M}_{\boldsymbol{f}^{(\boldsymbol{X})}, \infty}$ such that, for all $\boldsymbol{\sigma} \in S_{[\boldsymbol{n}]}$, $\boldsymbol{M} \in \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})}, B}$, and all non-increasing vectors $\boldsymbol{a}^{(X_1)} \in \mathbb{R}_+^{[n_1]}, \ldots, \boldsymbol{a}^{(X_d)} \in \mathbb{R}_+^{[n_d]}$:

$$(\boldsymbol{M} \circ \boldsymbol{\sigma}) \cdot \boldsymbol{a}^{(X_1)} \cdots \boldsymbol{a}^{(X_d)} \leq \boldsymbol{C} \cdot \boldsymbol{a}^{(X_1)} \cdots \boldsymbol{a}^{(X_d)} \tag{13}$$

In the rest of this section we describe the solution $\boldsymbol{C}$. If all entries in $\boldsymbol{C}$ are ≥ 0 and $\leq B$, then $\boldsymbol{C} \in \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})}, B}$ and, by setting $\boldsymbol{M} \stackrel{\text{def}}{=} \boldsymbol{C}$ and $\boldsymbol{\sigma} \stackrel{\text{def}}{=}$ the identity permutations, the relation S represented by $\boldsymbol{M} \circ \boldsymbol{\sigma}$ maximizes $|Q_{\texttt{star}}|$, achieving our goal. But, somewhat surprisingly, we found that sometimes this worst-case $\boldsymbol{C}$ has entries $> B$ or < 0, yet it still achieves our goal of a tight upper bound for $|Q_{\texttt{star}}|$. This is why we allow $\boldsymbol{C} \in \mathcal{M}_{\boldsymbol{f}^{(\boldsymbol{X})}, \infty}$.

Let Δ_Z denote the *discrete derivative* of an $\boldsymbol{X}$-tensor w.r.t. a variable $Z \in \boldsymbol{X}$, and Σ_Z denote the *discrete integral*. Formally, if $\boldsymbol{a} \in \mathbb{R}^{[n]}$ is a Z-vector, then, setting $a_0 \stackrel{\text{def}}{=} 0$:

$$\forall i \in [n]: \qquad (\Delta_Z \boldsymbol{a})_i \stackrel{\text{def}}{=} a_i - a_{i-1} \qquad (\Sigma_Z \boldsymbol{a})_i = \sum_{j=1,i} a_j \tag{14}$$

Notice that:

$$\Sigma_Z(\Delta_Z \boldsymbol{a}) = \Delta_Z(\Sigma_Z \boldsymbol{a}) = \boldsymbol{a} \qquad \texttt{SUM}_Z(\Delta_Z \boldsymbol{a}) = a_n \tag{15}$$

The subscript in Δ, Σ indicates on which variable they act. For example, if $\boldsymbol{M}$ is an XYZ-tensor, then $(\Delta_Y \boldsymbol{M})_{xyz} \stackrel{\text{def}}{=} M_{xyz} - M_{x(y-1)z}$. One should think of the three operators $\Delta_X, \Sigma_X, \texttt{SUM}_X$ as analogous to the continuous operators $\frac{d\cdots}{dx}$, $\int \cdots dx$, $\int_0^n \cdots dx$.

▶ **Definition 3.1.** The *value* tensor, $\boldsymbol{V}^{\boldsymbol{f}^{(\boldsymbol{X})},B} \in \mathbb{R}_+^{[\boldsymbol{n}]}$, is defined by the following linear optimization problem:

$$\forall \boldsymbol{m} \in [\boldsymbol{n}]: \qquad V_{\boldsymbol{m}}^{\boldsymbol{f}^{(\boldsymbol{X})},B} \stackrel{\text{def}}{=} \text{Maximize: } \sum_{\boldsymbol{s} \le \boldsymbol{m}} M_{\boldsymbol{s}} \qquad (16)$$

$$\text{Where: } \boldsymbol{M} \in \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})},B}$$

The *worst-case* tensor, $\boldsymbol{C}^{\boldsymbol{f}^{(\boldsymbol{X})},B} \in \mathbb{R}^{[\boldsymbol{n}]}$, is defined as:

$$\boldsymbol{C}^{\boldsymbol{f}^{(\boldsymbol{X})},B} \stackrel{\text{def}}{=} \Delta_{X_1} \cdots \Delta_{X_d} \boldsymbol{V}^{\boldsymbol{f}^{(\boldsymbol{X})},B} \qquad (17)$$

We will drop the superscripts when clear from the context, and write simply $\boldsymbol{V}, \boldsymbol{C}$. Our main result in this section is:

▶ **Theorem 3.2.** *Let $\boldsymbol{f}^{(\boldsymbol{X})}, B$ be given as above, and let $\boldsymbol{V}, \boldsymbol{C}$ defined by* (16)-(17)*. Then:*

1. *$\boldsymbol{C}$ is a solution to Problem 2, i.e. $\boldsymbol{C} \in \mathcal{M}_{\boldsymbol{f}^{(\boldsymbol{X})},\infty}$ and it satisfies Eq.* (13)*. Furthermore, it is tight in the following sense: there exists a tensor $\boldsymbol{M} \in \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})},B}$ and non-increasing vectors $\boldsymbol{a}^{(p)} \in \mathbb{R}_+^{[n_p]}$, $p = 1, d$, such that inequality* (13) *(with $\boldsymbol{\sigma}$ the identity) is an equality.*
2. *If there exists any solution $\boldsymbol{C}' \in \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})},B}$ to Problem 2, then $\boldsymbol{C}' = \boldsymbol{C}$.*
3. *When the number of dimensions is $d = 2$ then $\boldsymbol{C}$ is integral and non-negative. If $d \ge 3$, $\boldsymbol{C}$ may have negative entries.*
4. *If $B < \infty$, then $\boldsymbol{C}$ may not be consistent with B, even if $d = 2$.*
5. *For any non-increasing vectors $\boldsymbol{a}^{(X_p)} \in \mathbb{R}_+^{[n_p]}$, $p = 2, d$, the vector $\boldsymbol{C} \cdot \boldsymbol{a}^{(X_2)} \cdots \boldsymbol{a}^{(X_d)}$ is in $\mathbb{R}_+^{[n_1]}$ and non-increasing.*
6. *Assume $B = \infty$. Then the following holds:*

$$\forall \boldsymbol{m} \in [\boldsymbol{n}]: \qquad V_{\boldsymbol{m}} = \min\left(F_{m_1}^{(X_1)}, \ldots, F_{m_d}^{(X_d)}\right) \qquad (18)$$

where $F_r^{(X_p)} \stackrel{def}{=} \sum_{j \le r} f_j^{(X_p)}$ is the CDF associated to the PDF $\boldsymbol{f}^{(X_p)}$, for $p = 1, d$. Moreover, $\boldsymbol{C}$ can be computed by Algorithm 1, which runs in time $\mathbf{O}(\sum_p n_p)$. This further implies that $\boldsymbol{C} \ge 0$, in other words $\boldsymbol{C} \in \mathcal{M}^+_{\boldsymbol{f}^{(\boldsymbol{X})},\infty}$.

Algorithm 1 Efficient construction of $\boldsymbol{C}$ when $B = \infty$.

$\forall p = 1, d : s_p \leftarrow 1; \quad \boldsymbol{C} = 0;$
while $\forall p : s_p < n_p$ **do**
  $p_{min} \leftarrow \arg\min_p(f_{s_p}^{(X_p)}) \qquad d_{min} \leftarrow \min_p(f_{s_p}^{(X_p)})$
  $C_{s_1,\ldots,s_d} \leftarrow d_{min}$
  $\forall p = 1, d : f_{s_p}^{(X_p)} \leftarrow f_{s_p}^{(X_p)} - d_{min}$
  $s_{p_{min}} \leftarrow s_{p_{min}} + 1$
end while
return C

In a nutshell, the theorem asserts that the tensor $\boldsymbol{C}$ defined in (17) is the optimal solution to Problem 2; this is stated in item 1. Somewhat surprisingly, $\boldsymbol{C}$ may be inconsistent w.r.t. B, and may even be negative. When that happens, then, by item 2, no consistent solution exists to Problem 2, hence we have to make do with $\boldsymbol{C}$. In that case $\boldsymbol{C}$ may not represent a traditional bag S, for example if it has entries < 0. However, this will not be a problem for computing the degree sequence bound in Sec. 4, because all we need is to compute the

product $\boldsymbol{C} \cdot \boldsymbol{a}^{(X_2)} \cdots \boldsymbol{a}^{(X_d)}$, which we need to be non-negative, and non-increasing: this is guaranteed by item 5. The last item gives more insight into $\boldsymbol{V}$ and, by extension, into $\boldsymbol{C}$. Recall that $V_{\boldsymbol{m}}$, defined by (16), is the largest possible sum of values of a consistent $m_1 \times m_2 \times \cdots \times m_d$ tensor $\boldsymbol{M}$. Since the sum in each hyperplane $X_1 = r$ of $\boldsymbol{M}$ is $\leq f_r^{(X_1)}$, it follows that $\sum_{\boldsymbol{s} \leq \boldsymbol{m}} M_{\boldsymbol{s}} \leq \sum_{r=1,m_1} f_r^{(X_1)} \stackrel{\text{def}}{=} F_{m_1}^{(X_1)}$. Repeating this argument for each dimension X_p implies that $V_{\boldsymbol{m}} \leq \min_{p=1,d}(F_{m_p}^{(X_p)})$. Item 6 states that this becomes an equality, when $B = \infty$.

▶ **Example 3.3.** Suppose that we want to maximize $\boldsymbol{a}^T \cdot \boldsymbol{M} \cdot \boldsymbol{b}$, where $\boldsymbol{M}$ is a 3×4 matrix with degree sequences $\boldsymbol{f} = (6, 3, 1)$ and $\boldsymbol{g} = (4, 3, 2, 1)$; assume $B = \infty$. The vectors $\boldsymbol{a}, \boldsymbol{b}$ are non-negative and non-increasing, but otherwise unknown. The theorem asserts that this product is maximized by the worst-case matrix $\boldsymbol{C}$. We show here the matrices $\boldsymbol{C}$ and $\boldsymbol{V}$ defined by (16) and (17), together with degree sequences $\boldsymbol{f}, \boldsymbol{g}$ next to $\boldsymbol{C}$, and the cumulative sequences $\boldsymbol{F} = \Sigma\boldsymbol{f}, \boldsymbol{G} = \Sigma\boldsymbol{g}$ next to $\boldsymbol{V}$:

$$\boldsymbol{C} = \begin{array}{c} \\ 6 \\ 3 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 4 & 3 & 2 & 1 \end{array} \\ \begin{pmatrix} 4 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad \boldsymbol{V} = \begin{array}{c} \\ 6 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccc} 4 & 7 & 9 & 10 \end{array} \\ \begin{pmatrix} 4 & 6 & 6 & 6 \\ 4 & 7 & 9 & 9 \\ 4 & 7 & 9 & 10 \end{pmatrix} \end{array}$$

We can check that $V_{m_1 m_2} = \min(F_{m_1}, G_{m_2})$; for example $V_{31} = \min(10, 4) = 4$. The worst-case matrix $\boldsymbol{C}$ is defined as the second discrete derivative of $\boldsymbol{V}$, more precisely $C_{m_1 m_2} = V_{m_1 m_2} - V_{m_1-1,m_2} - V_{m_1,m_2-1} + V_{m_1-1,m_2-1}$. Alternatively, $\boldsymbol{C}$ can be computed greedily, using Algorithm 1: start with $C_{11} \leftarrow \min(f_1, g_1) = 4$, decrease both f_1, g_1 by 4, set the rest of column 1 to 0 (because now $g_1 = 0$) and continue with C_{12}, etc. Another important property, which we will prove below in the Appendix (Eq. 35 [4]), is that, for all m_1, m_2, $\sum_{i \leq m_1, j \leq m_2} C_{ij} = V_{m_1 m_2}$; for example $\sum_{i \leq 2, j \leq 3} C_{ij} = 4 + 2 + 1 + 2 = 9 = V_{23}$.

While the proof of Theorem 3.2 provides interesting insight into the structure of the degree sequence bound, it is not necessary for understanding the remainder of the paper and requires the introduction of additional notation and machinery. Therefore, for the sake of space and clarity, we omit it from the main text and instead include a proof of each item in the appendix Section A.2 [4].

4 The Berge-Acyclic Query

We now turn to the general problem 1. Fix a Berge-acyclic query Q with relations $\boldsymbol{R} \stackrel{\text{def}}{=} \boldsymbol{R}(Q)$, degree sequences $\boldsymbol{f}^{R,Z}$, and max tuple multiplicities $B^{(R)}$ as in problem 1.

4.1 The Degree Sequence Bound

▶ **Theorem 4.1.** *For any tensors $\boldsymbol{M}^{(R)} \in \mathcal{M}^+_{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}}$ and permutations $\boldsymbol{\sigma}^{(R)}$, for $R \in \boldsymbol{R}$, the following holds:*

$$\mathit{SUM}_{\boldsymbol{X}} \left(\bigotimes_{R \in \boldsymbol{R}} (\boldsymbol{M}^{(R)} \circ \boldsymbol{\sigma}^{(R)}) \right) \leq \mathit{SUM}_{\boldsymbol{X}} \left(\bigotimes_{R \in \boldsymbol{R}} \boldsymbol{C}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}} \right) \stackrel{def}{=} DSB(Q) \tag{19}$$

where $\boldsymbol{C}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}}$ is the worst-case tensor from Def. 3.1.

The theorem simply says that the upper bound to the query Q can be computed by evaluating Q on the worst case instances, represented by the worst case tensors $\boldsymbol{C}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}}$. We call this quantity the *degree sequence bound* and denote it by $DSB(Q)$. When all max

Algorithm 2 Computing $DSB(Q) = \texttt{SUM}_{\boldsymbol{X}}\left(\bigotimes_{R\in\boldsymbol{R}} \boldsymbol{C}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)},B^{(R)}}\right)$.

for each variable $X \in \boldsymbol{X}$ and non-root relation $R \in \boldsymbol{R}$, $R \neq \texttt{root}$, in bottom-up order **do**
  $\boldsymbol{a}^{(X)} \stackrel{\text{def}}{=} \bigotimes_{R\in\text{children}(X)} \boldsymbol{w}^{(R)}$ // element-wise product
  $\boldsymbol{w}^{(R)} \stackrel{\text{def}}{=} \boldsymbol{C}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)},B^{(R)}} \cdot \boldsymbol{a}^{(X_2)} \cdots \boldsymbol{a}^{(X_k)}$ // where $\boldsymbol{X}_R = (X_1, \ldots, X_k)$, $X_1 = \texttt{parent}(R)$
end for
return $C^{\boldsymbol{f}^{(\texttt{root},\boldsymbol{X}_{\texttt{ROOT}})},B^{(\texttt{ROOT})}} \cdot \boldsymbol{a}^{(X_1)} \cdot \boldsymbol{a}^{(X_2)} \cdots \boldsymbol{a}^{(X_k)}$

tuple multiplicities $B^{(R)}$ are ∞, then the bound is tight, because in that case every worst-case tensor $\boldsymbol{C}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)},\infty}$ is in $\mathcal{M}^{+}_{\boldsymbol{f}^{(R,\boldsymbol{X}_R)},\infty}$ (by Th. 3.2 item 6); otherwise the bound may not be tight, but it is locally tight, in the sense of Th. 3.2 item 1.

Before we sketch the main idea of the proof, we note that an immediate consequence is that the degree sequence bound can be computed using a special case of the FAQ algorithm [12]. We describe this briefly in Algorithm 2. Recall that the incidence graph of Q is a tree T. Choose an arbitrary relation $\texttt{ROOT} \in \boldsymbol{R}(Q)$ and designate it as root, then make T a directed tree by orienting all its edges away from the root. Denote by $\texttt{parent}(R) \in \boldsymbol{X}_R$ the parent node of a relation $R \neq \texttt{ROOT}$, associate an X-vector $\boldsymbol{a}^{(X)}$ to each variable X, and a $\texttt{parent}(R)$-vector $\boldsymbol{w}^{(R)}$ to each relation name R, then compute these vectors by traversing the tree bottom-up, as shown in Algorithm 2. Notice that, when X is a leaf variable, then $\text{children}(X) = \emptyset$ and $\boldsymbol{a}^{(X)} = (1, 1, \ldots, 1)^T$; similarly, if $R(X)$ is leaf relation of arity 1 with variable X, then $\boldsymbol{w}^{(R)}$ is the degree sequence of its variable, because $\boldsymbol{w}^{(R)} = \boldsymbol{C}^{(\boldsymbol{f}^{(R,X)},B^{(R)})} = \boldsymbol{f}^{(R,X)}$. We provide an example in [4], Appendix A.3. It follows:

▶ **Corollary 4.2.** *The degree sequence bound $DSB(Q)$ can be computed in time polynomial in the size of the largest domain (data complexity).*

In the rest of this section we sketch the proof of Theorem 4.1, mostly to highlight the role of item 5 of Theorem 3.2, and defer the formal details to Appendix A.3 [4]. Fix tensors $\boldsymbol{M}^{(R)}$ and permutations $\boldsymbol{\sigma}^{(R)}$, for each $R \in \boldsymbol{R}$. Choose one relation, say $S \in \boldsymbol{R}$, assume it has k variables $X_1, \ldots, X_k$, then write the LHS of (19) as:

$$\texttt{SUM}_{\boldsymbol{X}_S}\left(\left(\boldsymbol{M}^{(S)} \circ \boldsymbol{\sigma}^{(S)}\right) \otimes \boldsymbol{b}_1 \otimes \cdots \otimes \boldsymbol{b}_k\right) \tag{20}$$

where each $\boldsymbol{b}_p$ is a tensor expression sharing only variable X_p with S, where we sum out all variables except X_p (using Eq. (5)). Compute the vectors $\boldsymbol{b}_p$ first, sort them in non-decreasing order, let τ_p be the permutation that sorts $\boldsymbol{b}_p$, and $\boldsymbol{\tau} \stackrel{\text{def}}{=} (\tau_1, \ldots, \tau_k)$. Then (20) equals:

$$\texttt{SUM}_{\boldsymbol{X}_S}\left(\left(\boldsymbol{M}^{(S)} \circ \boldsymbol{\sigma}^{(R)} \circ \boldsymbol{\tau}\right) \otimes (\boldsymbol{b}_1 \circ \tau_1) \otimes \cdots \otimes (\boldsymbol{b}_k \circ \tau_k)\right) \tag{21}$$

because sums are invariant under permutations. Since each $\boldsymbol{b}_p \circ \tau_p$ is sorted, by item 1 of Theorem 3.2, the expression above is $\leq$ to the expression obtained by replacing $\boldsymbol{M}^{(S)} \circ \boldsymbol{\sigma}^{(S)} \circ \boldsymbol{\tau}$ with the worst-case tensor $\boldsymbol{C}^{f^{(S,\boldsymbol{X}_S)},B^{(S)}}$. Thus, every tensor could be replaced by the worst-case tensor, albeit at the cost of applying some new permutations τ_p to other expressions. To avoid introducing these permutations, we proceed as follows. We choose an orientation of the tree T, as in Algorithm 2, then prove inductively, bottom-up the tree, that each tensor $\boldsymbol{M} \circ \boldsymbol{\sigma}$ can be replaced by the worst-case tensor $\boldsymbol{C}$ without decreasing the LHS of (19), *and* that the resulting vector (in the bottom-up computation) is sorted. To prove this, we re-examine Eq. (20), assuming X_1 is the parent variable of S. By induction, all the tensors occurring in $\boldsymbol{b}_2, \ldots, \boldsymbol{b}_k$ have already been replaced with worst-case tensors, *and* their results

are non-increasing vectors. Then, in Eq. (21) it suffices to apply the permutation τ to the parent expression $\boldsymbol{b}_1$ (which still has the old tensors $\boldsymbol{M} \circ \boldsymbol{\sigma}$), use item 1 of Theorem 3.2 to replace $\boldsymbol{M}^{(S)} \circ \boldsymbol{\sigma}^{(S)} \circ \boldsymbol{\tau}$ by $\boldsymbol{C}^{f^{(S,\boldsymbol{X}_S)},B^{(S)}}$, and, finally, use item 5 of Theorem 3.2 to prove that the result returned by the node S is a non-decreasing vector, as required.

4.2 Connection to the AGM and Polymatroid Bounds

We prove now that $DSB(Q)$ is always below the AGM [1] and the polymatroid bounds [14, 18].

The AGM bound is expressed in terms of the cardinalities of the relations. For each relation R, let N_R be an upper bound on its cardinality. Then the AGM bound is $AGM(Q) \stackrel{\text{def}}{=} \min_{\boldsymbol{w}} \prod_R N_R^{w_R}$, where the vector $\boldsymbol{w} = (w_R)_{R \in \boldsymbol{R}}$ ranges over the fractional edge covers of the hypergraph associated to Q. If a database instance satisfies $|R| \leq N_R$ for all R, then the size of the query is $|Q| \leq AGM(Q)$, and this bound is tight, i.e. there exists an instance for which we have equality.

The polymatroid bound uses both the cardinality constraints N_R and the maximum degrees. The general bound in [14] considers maximum degrees for any subset of variables, but throughout this paper we restrict to degrees of single variables, in which case the polymatroid bound is expressed in terms of the quantities N_R and $f_1^{(R,X)}$, one for each relation R and each of its variables X. The AGM bound is the special case when $f_1^{(R,X)} = N_R$ for all R. We review the general definition of the polymatroid bound in [4], Appendix A.4, but will mention that no closed formula is known for polymatroid bound, similar to the AGM bound. We give here the first such closed formula, for the case of Berge-acyclic queries. Let Q be a Berge-acyclic query with incidence graph T (which is a tree). Choose an arbitrary relation $\texttt{ROOT} \in \boldsymbol{R}(Q)$ to designate as the root of T, and for each other relation R, denote by $Z_R \stackrel{\text{def}}{=} \texttt{parent}(R)$, i.e. its unique variable pointing up the tree. Denote by:

$$PB(Q, \texttt{ROOT}) \stackrel{\text{def}}{=} N_{\texttt{ROOT}} \prod_{R \neq \texttt{ROOT}} f_1^{(R,Z_R)} \tag{22}$$

One can immediately check that the query answer on any database instance consistent with the statistics satisfies $|Q| \leq PB(Q, \texttt{ROOT})$. A *cover* of Q is set $\boldsymbol{W} = \{Q_1, Q_2, \ldots, Q_m\}$, for some $m \geq 1$, where each Q_i is a connected subquery of Q, and each variable of Q occurs in at least one Q_i, and we denote by:

$$PB(\boldsymbol{W}) \stackrel{\text{def}}{=} \prod_{i=1,m} \min_{\texttt{ROOT} \in \boldsymbol{R}(Q_i)} PB(Q_i, \texttt{ROOT}_i) \tag{23}$$

Since $|Q| \leq |Q_1| \cdot |Q_2| \cdots |Q_m|$, we also have $|Q| \leq PB(\boldsymbol{W})$. We prove in [4], Appendix A.4:

▶ **Theorem 4.3.** *The polymatroid bound of a Berge-acyclic query Q is $PB(Q) \stackrel{def}{=} \min_{\boldsymbol{W}} PB(\boldsymbol{W})$, where $\boldsymbol{W}$ ranges over all covers.*

▶ **Example 4.4.** Let $Q = R(X,Y), S(Y,Z), T(Z,U), K(U,V)$. Then $PB(Q,S) = f_1^{(R,Y)} N_S f_1^{(T,Z)} f_1^{(K,U)}$, $PB(\{R, TK\}) = N_R \cdot \min\left(N_T f^{(K,U)}, f^{(T,U)} N_k\right)$, and $PB(\{R,T,K\}) = N_R N_T N_K$.

If we restrict the formula to the AGM bound, i.e. all max degrees are equal to the cardinalities, $f_1^{(R,X)} = N_R$, then Eq. (22) becomes $\prod_{R \in \boldsymbol{R}(Q)} N_R$, while the polymatroid bound (23) becomes $\min_{\boldsymbol{W}} \prod_{R \in \boldsymbol{W}} N_R$, where $\boldsymbol{W}$ ranges over integral covers of Q. In particular, the AGM bound of a Berge-acyclic query can be obtained by restricting to integral edge covers, although this property fails for α-acylic queries. For example, consider the query

$R(X,Y), S(Y,Z), T(Z,X), K(X,Y,Z)$; when $|R| = |S| = |T| = |K|$ then the AGM bound is obtained by the edge cover $0, 0, 0, 1$, but when $|R| = |S| = |T| \ll |K|$ one needs the fractional cover $1/2, 1/2, 1/2, 0$. Next, we prove next that the degree sequence bound is always better.

▶ **Lemma 4.5.**
(1) *For any choice of root relation,* $\texttt{ROOT} \in \boldsymbol{R}(Q)$*:* $DSB(Q) \leq PB(Q, \texttt{ROOT})$.
(2) *For any cover* $Q_1, \ldots, Q_m$ *of* Q, $DSB(Q) \leq DSB(Q_1) \cdots DSB(Q_m)$

Proof. (1) Referring to Algorithm 2, we prove by induction on the tree that, for all $R \neq \texttt{ROOT}$, and every index i, $w_i^{(R)} \leq \prod_{S \in \texttt{tree}(R)} f_1^{(S,Z_S)}$. In other words, each element of the vector $\boldsymbol{w}^{(R)}$ is $\leq$ the product of all max degrees in the subtree rooted at R. Assuming this holds for all children of R, consider the definition of $\boldsymbol{w}^{(R)}$ in Algorithm 2. By induction hypothesis, for each vector $\boldsymbol{a}^{(X_p)}$ we have $a_{i_p}^{(X_p)} \leq \prod_{S \in \texttt{tree}(X_p)} f_1^{(S,Z_S)}$, a quantity that is independent of the index i_p, and therefore we obtain the following:

$$w_{i_1}^{(R)} = \left(C^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}} \cdot \boldsymbol{a}^{(X_2)} \cdots \boldsymbol{a}^{(X_k)} \right)_{i_1} \leq \left(\sum_{i_2 i_3 \cdots i_k} C_{i_1 i_2 i_3 \cdots i_k}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}} \right) \cdot \prod_{S \in \texttt{tree}(R), S \neq R} f_1^{(S,Z_S)}$$

and we use the fact that $\sum_{i_2 i_3 \cdots i_k} C_{i_1 i_2 \cdots i_k}^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}} \leq f_{i_1}^{(R,X_1)}$ because, by Theorem 3.2 item 1, $C^{\boldsymbol{f}^{(R,\boldsymbol{X}_R)}, B^{(R)}}$ is consistent with the degree sequence $f_1^{(R,X_1)}$, and, finally, $f_{i_1}^{(R,X_1)} \leq f_1^{(R,X_1)}$. This completes the inductive proof. The algorithm returns $C^{\boldsymbol{f}^{(\texttt{root},\boldsymbol{X}_{\texttt{ROOT}})}, B^{(\texttt{ROOT})}} \cdot \boldsymbol{a}^{(X_1)} \cdot \boldsymbol{a}^{(X_2)} \cdots \boldsymbol{a}^{(X_k)} \leq \texttt{SUM}(C^{\boldsymbol{f}^{(\texttt{root},\boldsymbol{X}_{\texttt{ROOT}})}, B^{(\texttt{ROOT})}}) \cdot \prod_{R \neq \texttt{ROOT}} f_1^{(R,Z_R)} \leq |\texttt{ROOT}| \cdot \prod_{R \neq \texttt{ROOT}} f_1^{(R,Z_R)}$, which is $= PB(Q, \texttt{ROOT})$, as required.

(2) We prove the statement only for $m = 2$ (the general case is similar) and show that $DSB(Q) \leq DSB(Q_1) \cdot DSB(Q_2)$. Since DSB is the query answer on the worst case instance, we need to show that $|Q_1 \bowtie Q_2| \leq |Q_1| \cdot |Q_2|$. This is not immediately obvious because the worst case instance may have negative multiplicities. Let X be the unique common variable of Q_1, Q_2, and let $\boldsymbol{a}$, $\boldsymbol{b}$ be the X-vectors representing the results of Q_1 and Q_2 respectively. It follows from Theorem 3.2 item 5 that $\boldsymbol{a}, \boldsymbol{b}$ are non-negative, therefore, $|Q| = \sum_i a_i b_i \leq (\sum_i a_i)(\sum_i b_i) = |Q_1| \cdot |Q_2|$. ◀

Our discussion implies:

▶ **Theorem 4.6.** *Let* Q *be a Berge-acyclic query. We denote by* $DSB(Q, \boldsymbol{f}, \boldsymbol{B})$ *the DSB computed on the statistics* $\boldsymbol{f} \stackrel{def}{=} (\boldsymbol{f}^{R,Z})_{R \in \boldsymbol{R}(Q), Z \in \boldsymbol{X}_R}$ *and* $\boldsymbol{B} \stackrel{def}{=} (B^{(R)})_{R \in \boldsymbol{R}(Q)}$*. Then:*

$$|Q| \leq DSB(Q, \boldsymbol{f}, \boldsymbol{1}) \leq DSB(Q, \boldsymbol{f}, \boldsymbol{B}) \leq DSB(Q, \boldsymbol{f}, \boldsymbol{\infty}) \leq PB(Q) \leq AGM(Q) \tag{24}$$

where $|Q|$ *is the answer to the query on an database instance consistent with the given statistics.*

Recall that both AGM and PB bounds are defined over set semantics only. While the AGM bound is tight, the PB bound is known to not be tight in general, and it is open whether it is tight for Berge-acyclic queries. Our degree sequence bound under either set or bag semantics improves over PB and, in the case of bag semantics ($B = \infty$) DSB is tight.

5 Functional Representation

A degree sequence requires, in general, $\Omega(n)$ space, where $n = \max_{X \in \boldsymbol{X}} n_X$ is the size of the largest domain, while cardinality estimators require sublinear space and time. However, a degree sequence can be *represented* compactly, using a staircase function as illustrated in

Fig. 1. In this section we show how the degree sequence bound, DSB, be approximated in quasi-linear time in the size of the functional representation. We call this approximate bound FDSB, show that DSB $\leq$ FDSB $\leq PB$, and show that the staircase functions can be further compressed, allowing a tradeoff between the memory size and computation time on one hand, and accuracy of the FDSB on the other hand. We restrict our discussion to $B^{(R)} = \infty$.

In this section we denote a vector element by $F(i)$ rather than F_i. For a non-decreasing vector $\boldsymbol{F} \in \mathbb{R}_+^{[n]}$, we denote by $\boldsymbol{F}^{-1} : \mathbb{R}_+ \to \mathbb{R}_+$ any function satisfying the following, for all v, $0 \leq v \leq F(n)$: if $F(i) < v$ then $i < F^{-1}(v)$, and if $F(i) > v$ then $i > F^{-1}(v)$. Such a function always exists[5], but is not unique. Then:

▶ **Lemma 5.1.** *Let $\boldsymbol{F}_1 \in \mathbb{R}_+^{[n_1]}, \ldots, \boldsymbol{F}_d \in \mathbb{R}_+^{[n_d]}$ be non-decreasing vectors satisfying $F_1(0) = 0$ and, for all $p = 1, d$, $F_1(n_1) \leq F_p(n_p)$. Let $a_1 \in \mathbb{R}_+^{[n_1]}, \ldots, a_d \in \mathbb{R}_+^{[n_d]}$ be non-increasing vectors. Denote by $\boldsymbol{C}, \boldsymbol{w}$ the following tensor and vector:*

$$C_{i_1 \cdots i_d} \stackrel{def}{=} \Delta_{i_1} \cdots \Delta_{i_d} \max(F_1(i_1), \ldots, F_d(i_d)) \tag{25}$$

$$w(i_1) \stackrel{def}{=} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} C_{i_1 \cdots i_d} \prod_{p \in [2,d]} a_p(i_p) \tag{26}$$

Then the following inequalities hold:

$$w(i_1) \geq (\Delta_{i_1} F_1(i_1)) \prod_{p \in [2,d]} a_p\left(\left\lfloor F_p^{-1}(F_1(i_1)) \right\rfloor + 1\right) \tag{27}$$

$$w(i_1) \leq (\Delta_{i_1} F_1(i_1)) \prod_{p \in [2,d]} a_p\left(\left\lceil F_p^{-1}(F_1(i_1 - 1)) \right\rceil\right) \tag{28}$$

We give the proof in Appendix [4]. The lemma implies that, in Algorithm 2, we can use inequality (28) to upper bound the computation $w^{(R)} = \boldsymbol{C} \cdot \boldsymbol{a}^{(X_2)} \cdots \boldsymbol{a}^{(X_k)}$. Indeed, in that case each $F_p(r) \stackrel{\text{def}}{=} \sum_{i=1,r} f_p(r)$ is the cdf of a degree sequence f_p, hence $F_p(0) = 0$ and $F_p(n_p) =$ the cardinality of R, while the tensor $\boldsymbol{C}$ is described in item 6 of Theorem 3.2, hence the assumptions of the lemma hold.

We say that a vector $\boldsymbol{f} \in \mathbb{R}_+^n$ is *represented* by a function $\hat{f} : \mathbb{R}_+ \to \mathbb{R}_+$ if $f(i) = \hat{f}(i)$ for all $i = 1, n$. A function $\hat{f}$ is a *staircase function with s steps*, in short an *s-staircase*, if there exists dividers $m_0 \stackrel{\text{def}}{=} 0 < m_1 < \cdots < m_s \stackrel{\text{def}}{=} n$ such that $\hat{f}(x)$ is a nonnegative constant on each interval $\{x \mid m_{q-1} < x \leq m_q\}$, $q = 1, s$. The sum or product of an s_1-staircase with an s_2-staircase is an $(s_1 + s_2)$-staircase. We denote the summation of a staircase $\hat{f}(x)$ as $\hat{F}(x) = \int_0^x \hat{f}(t)dt$ which is then an increasing piecewise-linear function. Its standard inverse $\hat{F}^{-1} : \mathbb{R}_+ \to \mathbb{R}_+$ is also increasing and piecewise-linear. If $\hat{F}$ represents the vector F, then $\hat{F}^{-1}$ is an inverse F^{-1} of that vector (as discussed above).

Fix a Berge-acyclic query Q, and let each degree sequence $\boldsymbol{f}^{(R,Z)}$ be represented by some $s_{R,Z}$-staircase $\hat{f}^{R,Z}$, and we denote by $\hat{F}^{(R,Z)}$ its summation. Fix any relation $\texttt{ROOT} \in \boldsymbol{R}(Q)$ to designated as root. The *Functional Degree Sequence Bound at* ***ROOT***, $FDSB(Q, \texttt{ROOT})$, is the value returned by Algorithm 3. This algorithm is identical to Algorithm 2, except that it replaces both $\boldsymbol{w}^{(R)}$ with a functional upper bound justified by the inequality 28 of Lemma 5.1, and similarly for the returned result. All functions $\hat{\boldsymbol{a}}^{(X)}$ and $\hat{\boldsymbol{w}}^{(R)}$ are staircase functions, and can be computed in linear time, plus a logarithmic time need for a binary search to lookup a segment in a staircase. Using this, we prove the following in Appendix A.6 [4]:

[5] E.g. define it as follows: if $\exists i$ s.t. $F(i-1) < v < F(i)$ then set $F^{-1}(v) \stackrel{\text{def}}{=} i - 1/2$, otherwise set $F^{-1}(v) = i$ for some arbitrary i s.t. $F(i) = v$.

Algorithm 3 $FDSB(Q, \texttt{ROOT})$.

for each variable $X \in \boldsymbol{X}$ and non-root relation $R \in \boldsymbol{R}$, $R \neq \texttt{root}$, in bottom-up order **do**

$\hat{\boldsymbol{a}}^{(X)} \stackrel{\text{def}}{=} \bigotimes_{R \in \text{children}(X)} \hat{\boldsymbol{w}}^{(R)}$

$\forall i_1: \quad \hat{w}^{(R)}(i_1) \stackrel{\text{def}}{=} \left(\hat{f}^{(R,X_1)}(i_1)\right) \prod_{p \in [2,d]} a^{(X_p)} \left(\max(1, (\hat{F}^{(R,X_p)})^{-1}(\hat{F}^{(R,X_1)}(i_1 - 1)))\right)$

end for

return $\sum_{i=1,|\texttt{ROOT}|} \prod_{p=1,k} a^{(X_p)}(\max(1, (F^{\texttt{ROOT},X_p})^{-1}(i-1)))$

▶ **Theorem 5.2.**

(1) $FDSB(Q, \texttt{ROOT}) \geq DSB(Q)$.

(2) $FDSB(Q, \texttt{ROOT})$ *can be computed in time* $T_{FDSB} \stackrel{def}{=} \tilde{O}(m \cdot \sum_{R,Z}(arity(R) \cdot s_{R,Z}))$, *where* $\tilde{O}$ *hides a logarithmic term, and* $m = |\boldsymbol{R}(Q)|$ *is the number of relations in* Q.

The theorem says that $FDSB(Q, \texttt{ROOT})$ is still an upper bound on $|Q|$, and can be computed in quasi-linear time in the size of the functional representations of the degree sequences. Next, we check if $FDSB$ is below the polymatroid bound. Consider the computation of $\hat{w}^{(R)}(i_1)$ by the algorithm. On one hand $\hat{f}^{(R,X_1)}(i_1) \leq \hat{f}^{(R,X_1)}(1)$; on the other hand $a^{(X_p)}(\max(1, \ldots)) \leq a^{(X_p)}(1)$. This allows us to prove (inductively on the tree, in [4], Appendix A.7):

▶ **Lemma 5.3.** $FDSB(Q, \texttt{ROOT}) \leq PB(Q, \texttt{ROOT})$, *where* PB *is defined in* (22).

When we proved $DSB \leq PB$ in Lemma 4.5, we used two properties of DSB: $DSB(Q, \texttt{ROOT})$ is independent of the choice of $\texttt{ROOT}$, and $DSB(Q_1 \bowtie \cdots \bowtie Q_m) \leq DSB(Q_1) \cdots DSB(Q_m)$, for any cover $\boldsymbol{W} = \{Q_1, \ldots, Q_m\}$. Both hold because $DSB(Q)$ is standard query evaluation: it is independent of the query plan (i.e. choice of $\texttt{ROOT}$) and it can only increase if we remove join conditions. But $FDSB$ is no longer standard query evaluation and these properties may fail. For that reason we introduce a stronger functional degree sequence bound:

$$FDSB(Q) = \min_{\boldsymbol{W}} \prod_{i=1,m} \min_{ROOT \in \boldsymbol{R}(Q)} FDSB(Q_i, ROOT) \tag{29}$$

where $\boldsymbol{W}$ range over the covers of Q. We prove in Appendix A.6.1 [4]:

▶ **Theorem 5.4.** $FDSB(Q)$ *can be computed in time* $O(2^m \cdot (2^m + m \cdot T_{FDSB}))$ *(where* T_{FDSB} *is defined in Theorem 5.2).*

Mirroring our results from Theorem 4.6, we prove the following in Appendix A.8 [4]:

▶ **Theorem 5.5.** *Suppose* Q *is a Berge-acyclic query. Then the following hold:*

$$|Q| \leq FDSB(Q) \leq PB(Q) \leq AGM(Q) \tag{30}$$

Together, Theorems 5.4 and 5.5 imply that we can compute in quasi-linear time in the size of the representation an upper bound to the query Q that is guaranteed to improve over the polymatroid bound. In practice, we expect this bound to be significantly lower than the polymatroid bound, because it accounts for the entire degree sequence $\boldsymbol{f}$, not just f_1.

Finally, we show that one can tradeoff the size of the representation for accuracy, by simply choosing more coarse staircase approximations of the degree sequences. They only need to be non-increasing, and lie above the true degree sequences.

▶ **Theorem 5.6.** *Fix a query Q, let $\boldsymbol{f}^{(R,Z)}, B^{(R)}$ be statistics as in Problem 1, and let U be the cardinality bound defined by* (10)*. Let $\hat{\boldsymbol{f}}^{(R,Z)}, \hat{B}^{(R)}$ be a new set of statistics, and $\hat{U}$ the resulting cardinality bound. If $\boldsymbol{f}^{(R,\boldsymbol{X}_R)} \leq \hat{\boldsymbol{f}}^{(R,\boldsymbol{X}_R)}$ and $B^{(R)} \leq \hat{B}^{(R)}$ for all R, $Z \in X_R$, then $U \leq \hat{U}$.*

Proof. The proof follows immediately from the observation that the set of feasible solutions can only increase (see Def. 2.3): $\mathcal{M}^{+}_{\boldsymbol{f}^{(R,\boldsymbol{X}_R)},B^{(R)}} \subseteq \mathcal{M}^{+}_{\hat{\boldsymbol{f}}^{(R,\boldsymbol{X}_R)},\hat{B}^{(R)}}$. ◀

6 Conclusions

We have described the *degree sequence bound* of a conjunctive query, which is an upper bound on the size of its answer, given in terms of the degree sequences of all its attributes. Our results apply to Berge-acyclic queries, and strictly improve over previously known AGM and polymatroid bounds [1, 14]. On one hand, our results represent a significant extension, because they account for the full degree sequences rather than just cardinalities or just the maximum degrees. On the other hand, they apply only to a restricted class of acyclic queries, although, we argue, this class is the most important for practial applications. While the full degree sequence can be as large as the entire data, we also described how to approximate the cardinality bound very efficiently, using compressed degree sequences. Finally, we have argued for using the max tuple multiplicity for each relation, which can significantly improve the accuracy of the cardinality bound.

References

1 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 739–748. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.43.

2 Douglas Bauer, Haitze J Broersma, Jan van den Heuvel, Nathan Kahl, A Nevo, E Schmeichel, Douglas R Woodall, and Michael Yatauro. Best monotone degree conditions for graph properties: a survey. *Graphs and combinatorics*, 31(1):1–22, 2015.

3 Walter Cai, Magdalena Balazinska, and Dan Suciu. Pessimistic cardinality estimation: Tighter upper bounds for intermediate join cardinalities. In Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 18–35. ACM, 2019. doi:10.1145/3299869.3319894.

4 Kyle Deeds, Dan Suciu, Magda Balazinska, and Walter Cai. Degree sequence bound for join cardinality estimation. *arXiv preprint*, 2022. arXiv:2201.04166.

5 Ronald Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *J. ACM*, 30(3):514–550, 1983. doi:10.1145/2402.322390.

6 Amir Gilad, Shweta Patwa, and Ashwin Machanavajjhala. Synthesizing linked data under cardinality and integrity constraints. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 619–631. ACM, 2021. doi:10.1145/3448016.3457242.

7 Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. Size and treewidth bounds for conjunctive queries. *J. ACM*, 59(3):16:1–16:35, 2012. doi:10.1145/2220357.2220363.

8 Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 289–298. ACM Press, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109590.

9 S Louis Hakimi and Edward F Schmeichel. Graphs and their degree sequences: A survey. In *Theory and applications of graphs*, pages 225–235. Springer, 1978.

10 Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, et al. Cardinality estimation in dbms: A comprehensive benchmark evaluation. *arXiv preprint arXiv:2109.05877*, 2021.

11 Axel Hertzschuch, Claudio Hartmann, Dirk Habich, and Wolfgang Lehner. Simplicity done right for join ordering. In *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. www.cidrdb.org, 2021. URL: http://cidrdb.org/cidr2021/papers/cidr2021_paper01.pdf.

12 Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. FAQ: questions asked frequently. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 13–28. ACM, 2016. doi:10.1145/2902251.2902280.

13 Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. Computing join queries with functional dependencies. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 327–342. ACM, 2016. doi:10.1145/2902251.2902289.

14 Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 429–444. ACM, 2017. doi:10.1145/3034786.3056105.

15 Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? *Proc. VLDB Endow.*, 9(3):204–215, 2015. doi:10.14778/2850583.2850594.

16 Jie Liu, Wenqian Dong, Dong Li, and Qingqing Zhou. Fauce: Fast and accurate deep ensembles with uncertainty for cardinality estimation. *Proc. VLDB Endow.*, 14(11):1950–1963, 2021. URL: http://www.vldb.org/pvldb/vol14/p1950-liu.pdf, doi:10.14778/3476249.3476254.

17 Parimarjan Negi, Ryan C. Marcus, Andreas Kipf, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. Flow-loss: Learning cardinality estimates that matter. *Proc. VLDB Endow.*, 14(11):2019–2032, 2021. URL: http://www.vldb.org/pvldb/vol14/p2019-negi.pdf, doi:10.14778/3476249.3476259.

18 Hung Q. Ngo. Worst-case optimal join algorithms: Techniques, results, and open problems. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 111–124. ACM, 2018. doi:10.1145/3196959.3196990.

19 Yeonsu Park, Seongyun Ko, Sourav S. Bhowmick, Kyoungmin Kim, Kijae Hong, and Wook-Shin Han. G-CARE: A framework for performance benchmarking of cardinality estimation techniques for subgraph matching. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1099–1114. ACM, 2020. doi:10.1145/3318464.3389702.

20 Ji Sun, Guoliang Li, and Nan Tang. Learned cardinality estimation for similarity queries. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 1745–1757. ACM, 2021. doi:10.1145/3448016.3452790.

21 Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. Are we ready for learned cardinality estimation? *Proc. VLDB Endow.*, 14(9):1640–1654, 2021. URL: http://www.vldb.org/pvldb/vol14/p1640-wang.pdf, doi:10.14778/3461535.3461552.

22 Peizhi Wu and Gao Cong. A unified deep model of learning from both data and queries for cardinality estimation. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2009–2022. ACM, 2021. doi:10.1145/3448016.3452830.

23 Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. Neurocard: One cardinality estimator for all tables. *Proc. VLDB Endow.*, 14(1):61–73, 2020. doi:10.14778/3421424.3421432.

24 Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. FLAT: fast, lightweight and accurate method for cardinality estimation. *Proc. VLDB Endow.*, 14(9):1489–1502, 2021. URL: http://www.vldb.org/pvldb/vol14/p1489-zhu.pdf, doi:10.14778/3461535.3461539.