PANDAExpress: A Simpler and Faster PANDA Algorithm

MAHMOUD ABO KHAMIS, RelationalAI, USA
HUNG Q. NGO, RelationalAI, USA
DAN SUCIU, University of Washington, USA

PANDA is a powerful generic algorithm for answering conjunctive queries and disjunctive datalog rules given input degree constraints. In the special case of Boolean queries with only cardinality constraints, PANDA runs in $\tilde{O}(N^{\text{subw}})$-time, where N is the input size, and subw is the submodular width of the query, a notion introduced by Daniel Marx (JACM 2013) in the context of constraint satisfaction problems. When specialized to certain classes of sub-graph pattern finding problems, the $\tilde{O}(N^{\text{subw}})$ runtime matches the optimal runtime possible, modulo some conjectures in fine-grained complexity (Bringmann and Gorbachev (STOC 25)). The PANDA framework is much more general, as it handles arbitrary input degree constraints, which capture common statistics and integrity constraints used in relational database management systems, it works for queries with free variables, and for both conjunctive queries and disjunctive datalog rules.

The key weakness of PANDA is the large polylog(N)-factor hidden in the $\tilde{O}(\cdot)$ notation, making it completely impractical, and falling short of what is achievable with specialized algorithms. This paper resolves this weakness with two novel ideas. First, we prove a new probabilistic inequality that upper-bounds the output size of disjunctive datalog rules under arbitrary degree constraints. Second, the proof of this inequality directly leads to a new algorithm named PANDAExpress that is both *simpler* and *faster* than PANDA. A novel feature of PANDAExpress is a new partitioning scheme that uses arbitrary hyperplane cuts instead of axis-parallel hyperplanes used in PANDA. These hyperplanes are dynamically constructed based on data-skewness statistics carefully tracked throughout the algorithm's execution. As a result, PANDAExpress removes the polylog(N)-factor from the runtime of PANDA, matching the runtimes of intricate specialized algorithms, while retaining all its generality and power. As a bonus, we also show how PANDAExpress can handle ℓ_p-norm constraints, which generalize degree constraints.

CCS Concepts: • **Mathematics of computing → Probability and statistics**; **Information theory**; • **Information systems → Join algorithms**; • **Theory of computation → Database query processing and optimization (theory)**.

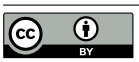
Additional Key Words and Phrases: PANDA; conjunctive queries; disjunctive datalog rules; submodular width; proof sequences; Shannon inequalities

ACM Reference Format:
Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2026. PANDAExpress: A Simpler and Faster PANDA Algorithm. *Proc. ACM Manag. Data* 4, 2 (PODS), Article 114 (May 2026), 26 pages. https://doi.org/10.1145/3801910

1 Introduction

Conjunctive query (CQ) evaluation is a fundamental problem in many areas of computer science, from traditional ones such as relational database management, graph analytics, sparse tensor kernels, constraint satisfaction, logic [1, 3, 22, 25], to other settings such as deciding query equivalence [14]

Authors' Contact Information: Mahmoud Abo Khamis, mahmoud.abokhamis@relational.ai, RelationalAI, Berkeley, CA, USA; Hung Q. Ngo, hung.ngo@relational.ai, RelationalAI, Berkeley, CA, USA; Dan Suciu, suciu@cs.washington.edu, University of Washington, Seattle, WA, USA.



ACM 2836-6573/2026/5-ART114
https://doi.org/10.1145/3801910

and equality saturation [40]. There has been a lot of work in both theory and practice to develop efficient query planning and evaluation techniques for evaluating CQs for at least 50 years [18, 34].

The past 15 years has witnessed an exciting new paradigm for optimizing and evaluating CQs, involving two key ideas. First, we can derive a non-trivial worst-case output size bound of a CQ, or more generally a disjunctive datalog rule, given commonly collected input statistics [4, 5, 10, 15, 20, 23]. Second, one can turn a *proof* of this bound into a query plan to answer the query efficiently [6, 32, 33, 37]. These new query plans are completely non-trivial and very different from the traditional query plans based on select-project-join trees [6, 31, 35].

These new query plans boil down to three basic operations: *partitioning* the data based on the frequencies of the values in some columns, *joining* two or more relations, and *projection*. Instances of this type of query plans can be traced back to a special case of CQ evaluation: the problem of finding a small (hyper)graph pattern inside a large graph [7, 10, 21, 24, 25, 27]. For example, Itai and Rodeh [27] described an $O(|E|^{3/2})$-time triangle detection algorithm by partitioning vertices into heavy and light based on degree threshold $|E|^{1/2}$. Alon, Yuster, and Zwick [8] generalized the idea to the k-cycle detection problem with runtime $O(|E|^{2-\frac{1}{\lceil k/2 \rceil}})$ by adjusting the threshold and distributing the computational load to different tree decompositions of the k-cycle graph. Recently, algorithms to find much more complex graph patterns were discovered by Bringmann and Gorbachev [12, 13]. While the previous algorithms partition the data into two classes, the algorithm in [13] partitions the data into a number of classes that depends on the graph pattern (the query), which is still $O(1)$ in the size of the input database.

In an RDBMS, queries and data statistics are more general than those in the graph-pattern finding problem. CQs have relations of arbitrary arities, the output has arbitrary free variables, and the data statistics can include not only relation sizes, but also "degree constraints" which bound the number of distinct values in some columns given fixed values in other columns [4].

The pioneering work of Marx [30] deals with Boolean CQs over relations with arbitrary arities, using multiple tree decompositions to load-balance. Let $\mathrm{subw}(Q)$ denote the *submodular width* of a Boolean query Q, then Marx's algorithm runs in time $\tilde{O}(N^{c\cdot \mathrm{subw}(Q)})$ to answer Q on any database of size N, and $c > 1$ is a constant. While Marx's algorithm does not handle degree constraints and arbitrary free variables, his notion of submodular width is very powerful. In addition to it being the optimal parameter for fixed-parameter tractability of CSP [30], it also is the precise *optimal* parameter for a class of sub-graph identification and listing queries in the fined-grained complexity setting, as shown very recently by Bringmann and Gorbachev [13].

The natural question arises whether any CQ (Boolean or not) under arbitrary degree constraints can be computed optimally by the same strategy of partitioning the data into $O(1)$ classes, based on degrees, and performing joins. A general algorithm called PANDA for computing CQs was described in [6], and it has the best known runtime for answering an arbitrary CQ using a combinatorial algorithm (i.e. without using Fast Matrix Multiplication), except for an undesired large polylogarithmic factor. This undesired factor is due to the fact that, at each partition step, PANDA needs to partition a relation into $\log N$ parts, and the number of partitioning steps is a function of the query. In particular, PANDA's runtime is $O\left(N^{\mathrm{subw}(Q)}\mathrm{polylog}(N) + |Q|\right)$, where $\mathrm{subw}(Q)$ is the submodular width of the query Q, N is the input size, and $|Q|$ is the output size.

The polylog factor means PANDA provides an unsatisfactory answer to our desire of understanding the complexity of query evaluation, despite its generality and power. Its shortcoming does not stop at combinatorial algorithms, but also extends to algorithms that use Fast Matrix Multiplication (FMM). All prior algorithms using FMM [8, 16, 17, 19] apply the same principle of partitioning the data into $O(1)$ classes, then using either joins or FMM in each class. In contrast, the generalization

of PANDA to using FMM described in [28] also partitions the data into polylogarithmic number of classes, falling short of the special cases described in the literature by a polylogarithmic factor.

This unsatisfactory state of the art naturally leads to the question whether it is possible to design an algorithm that shaves off PANDA's polylogarithmic factor while retaining all its generality. On a closer examination, we found that the specialized algorithms for graph-pattern finding in the literature [8, 13] not only partition the data into $O(1)$ parts at each step, but have another important common characteristic: their partitioning criterion consists of comparing a single degree to a pre-computed threshold ("heavy" or "light"). We call this an *axis-parallel* partitioning method; because, as we shall explain in Section 3, the strategy corresponds to partitioning the input relations using axis-parallel hyperplanes. PANDA also does an axis-parallel partitioning; it just uses many more bins. The question then becomes whether it is possible to design an algorithm that answers any CQ in submodular width time using only $O(1)$ axis-parallel parts at each step.

Contributions. This paper shows that the answer to the above question is likely to be negative, and describes instead a very simple algorithm that answers a CQ in time $O\left(N^{\mathrm{subw}(Q)} \log N + |Q|\right)$ by using partitions based on hyperplane cuts that are not necessarily axis-parallel. (The remaining log-factor represents a sorting step.)

Specifically, the main problem we consider is the worst-case-optimally answering of *Disjunctive Datalog Rules* (DDR), which are *the* main building block for answering CQs using the PANDA framework [5, 6]. DDRs capture the essence of data partitioning: the body of a DDR is a conjunction, the head is a disjunction of atoms, and the goal is to "partition" the outputs of the CQ among the head atoms so as to minimize the largest cardinality among the output relations.

The measuring stick for the notion of worst-case optimality [31] is a tight bound on the output size of a DDR given input degree constraints. Our *first contribution* is a new probabilistic inequality that can be used to prove the same tight output size bounds for DDRs as the one shown in [5, 6].

Our *second contribution* is a new algorithm, called PANDAExpress, derived from the proof of the probabilistic inequality. The algorithm answers DDRs in time $O((N + B) \log N)$ where N is the input size and B is the worst-case size bound of the DDR under the given degree constraints. As shown in [5, 6], this result immediately implies an algorithm for answering CQs in time $O\left(N^{\mathrm{subw}(Q)} \log N + |Q|\right)$, where $\mathrm{subw}(Q)$ is the submodular width of the query Q, and $|Q|$ is the output size. As can be seen in Algorithm 1, PANDAExpress is *breathtakingly simple*, which is another key takeaway from this work: it is not only faster but also much simpler than PANDA.

As a *third contribution*, we also show that the same algorithm can be used to handle ℓ_p-norm constraints [29], which generalize degree constraints. These constraints have been shown to be very useful in practice for query optimization [39].

The crucial insights are as follows. First, we show that $O(1)$ axis-parallel partitions are *insufficient* for optimality in the general case of DDRs. While this worked fine for all special cases studied in the literature [8, 12, 13, 27], it fails in the general case. Instead, PANDAExpress partitions the input and intermediate relations using $O(1)$ *arbitrary hyperplanes*. Second, the hyperplanes separating the data are also not statically determined. The algorithm collects and maintains data-skew statistics along the way, which are then used to determine the hyperplanes to partition the data. The dynamic partitioning strategy is designed to *load-balance* amongst sub-query plans in a fine grained manner. This level of precision was not possible in query plans that make use of only polymatroid-based proof sequences as in the original PANDA algorithm, because the polymatroids model the entropies which are too coarse to capture data-level skewness.

Outline. The rest of the paper is organized as follows. Section 2 briefly presents background knowledge needed for the paper. Section 3 presents examples and motivations for our work; in particular, it presents some examples where axis-parallel partitioning alone is insufficient. Section 4

states and proves our new probabilistic inequality. Section 5 presents the new PANDA algorithm to answer DDRs, based on the new inequality. Section 6 proves the correctness of the algorithm and analyzes its runtime. Section 7 explains how to use the algorithm for answering CQs in submodular width time. Section 8 shows how to use the algorithm to handle ℓ_p-norm constraints. We conclude in Section 9 with some final remarks and future directions.

2 Background

2.1 Polymatroids and Shannon (Flow) Inequalities

Let $\boldsymbol{V}$ be a set of variables. Let $\boldsymbol{X}, \boldsymbol{Y} \subseteq \boldsymbol{V}$ be disjoint sets of variables, then the pair $\delta = (\boldsymbol{Y}|\boldsymbol{X})$ is called a *monotonicity term*. A monotonicity term of the form $\delta = (\boldsymbol{Y}|\emptyset)$ is called an *unconditional monotonicity term*, in which case we simply write $\delta = \boldsymbol{Y}$. Let $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \subseteq \boldsymbol{V}$ be 3 disjoint sets of variables, then the triple $\sigma = (\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})$ is called a *submodularity term*. Let MON and SUB denote the sets of all monotonicity terms and submodularity terms over $\boldsymbol{V}$, respectively.

For any two sets $\boldsymbol{X}, \boldsymbol{Y}$ of variables, we write $\boldsymbol{XY}$ for $\boldsymbol{X} \cup \boldsymbol{Y}$. Given a function $h : 2^{\boldsymbol{V}} \to \mathbb{R}_+$, define

$$h(\delta) \stackrel{\text{def}}{=} h(\boldsymbol{XY}) - h(\boldsymbol{X}), \qquad \delta = (\boldsymbol{Y}|\boldsymbol{X}) \tag{1}$$

$$h(\sigma) \stackrel{\text{def}}{=} h(\boldsymbol{XY}) + h(\boldsymbol{XZ}) - h(\boldsymbol{XYZ}) - h(\boldsymbol{X}) = h(\boldsymbol{Y}|\boldsymbol{X}) - h(\boldsymbol{Y}|\boldsymbol{XZ}), \qquad \sigma = (\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X}) \tag{2}$$

The function h can also be viewed interchangeably as a vector $\boldsymbol{h} = (h(\boldsymbol{S}))_{\boldsymbol{S} \subseteq \boldsymbol{V}} \in \mathbb{R}_+^{2^{\boldsymbol{V}}}$. A function $h : 2^{\boldsymbol{V}} \to \mathbb{R}_+$ is called a *polymatroid* if it satisfies all the *elemental Shannon inequalities*:

$$h(\emptyset) = 0 \qquad h(\delta) \geq 0,\ \forall \delta \in \mathsf{MON} \qquad h(\sigma) \geq 0,\ \forall \sigma \in \mathsf{SUB}, \tag{3}$$

where $h(\delta) \geq 0$ is called the *monotonicity* and $h(\sigma) \geq 0$ the *submodularity* property.

Shannon inequalities (beyond the elemental ones) are all inequalities obtained by taking non-negative linear combinations of the elemental Shannon inequalities (3). In other words, Shannon inequalities are linear inequalities that hold for all polymatroids. We are interested in Shannon inequalities of a special form, called *Shannon-flow inequalities* in [5].

Definition 2.1. Let Σ denote a set of *unconditional* monotonicity terms, and Δ denote another set of monotonicity terms. Let $\boldsymbol{\lambda} = (\lambda_Z)_{Z \in \Sigma}$ and $\boldsymbol{w} = (w_\delta)_{\delta \in \Delta}$ be two sets of non-negative *rational* coefficients with $\|\boldsymbol{\lambda}\|_1 = 1$. A *Shannon-flow inequality* is a Shannon inequality of the form

$$\sum_{Z \in \Sigma} \lambda_Z \cdot h(Z) \leq \sum_{\delta \in \Delta} w_\delta \cdot h(\delta) \tag{4}$$

Below is an equivalent characterization of Shannon-flow inequalities:

Proposition 2.2 ([6])**.** *Inequality* (4) *holds for all polymatroids iff there are multisets $\mathcal{Z}$, $\mathcal{D}$, $\mathcal{M}$, $\mathcal{S}$ over base sets Σ, Δ,* MON, SUB *respectively, such that*

$$\sum_{Z \in \mathcal{Z}} h(Z) = \sum_{\delta \in \mathcal{D}} h(\delta) - \sum_{\mu \in \mathcal{M}} h(\mu) - \sum_{\sigma \in \mathcal{S}} h(\sigma) \tag{5}$$

holds as an identity over symbolic variables $h(\boldsymbol{X})$, $\emptyset \neq \boldsymbol{X} \subseteq \boldsymbol{V}$. Furthermore, the sizes of the multisets are functions of $\boldsymbol{\lambda}, \boldsymbol{w}$, and $|\boldsymbol{V}|$.

Definition 2.3. Given identity (5), we recover the corresponding Shannon-flow inequality (4) by dropping the negative terms on the right-hand side of (5),

$$\sum_{Z \in \mathcal{Z}} h(Z) \leq \sum_{\delta \in \mathcal{D}} h(\delta) \tag{6}$$

and dividing both sides by $|\mathcal{Z}|$. We refer to inequality (6) an *integral Shannon-flow inequality*, that is parameterized by the multisets $(\mathcal{Z}, \mathcal{D})$, and *witnessed* by the multisets $(\mathcal{M}, \mathcal{S})$.

The following two key results about (integral) Shannon-flow inequalities were shown in [5, 6]. They are both straightforward consequences of Proposition 2.2. See Appendix A for a self-contained proof of these results.

Theorem 2.4 (The Proof Sequence Construction [5, 6]). *Every integral Shannon-flow inequality* (6) *parameterized by* $(\mathcal{Z}, \mathcal{D})$ *with witness* $(\mathcal{M}, \mathcal{S})$ *can be proved by a sequence of steps*

$$\sum_{\delta\in\mathcal{D}} h(\delta) = \sum_{\delta\in\mathcal{D}_0} h(\delta) \geq \sum_{\delta\in\mathcal{D}_1} h(\delta) \geq \cdots \geq \sum_{\delta\in\mathcal{D}_k} h(\delta), \text{ with } \mathcal{D}_k \supseteq \mathcal{Z}$$

where $\sum_{\delta\in\mathcal{D}_i} h(\delta)$ *is turned into* $\sum_{\delta\in\mathcal{D}_{i+1}} h(\delta)$ *by applying exactly one of the following steps:*

- Submodularity: $h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{Y}|\boldsymbol{X}\boldsymbol{Z})$.
- Composition: $h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{X}\boldsymbol{Y})$.
- Decomposition: $h(\boldsymbol{X}\boldsymbol{Y}) \to h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{X})$.
- Monotonicity: $h(\boldsymbol{X}\boldsymbol{Y}) \to h(\boldsymbol{X})$.

Furthermore, $(\mathcal{Z}, \mathcal{D}_i)$ *is an integral Shannon-flow inequality witnessed by* $(\mathcal{M}_i, \mathcal{S}_i)$*, for all* $0 \leq i \leq k$*, and the number of steps is bounded by* $k \leq |\mathcal{D}| + |\mathcal{M}| + 3|\mathcal{S}|$.

Later, Fig. 2 has an example of a proof sequence for a Shannon-flow inequality from Eq. (21).

Lemma 2.5 (The Reset Lemma [6])**.** *Given an integral Shannon-flow inequality* (6) *parameterized by* $(\mathcal{Z}, \mathcal{D})$ *and witnessed by* $(\mathcal{M}, \mathcal{S})$*, let* $\boldsymbol{W} \in \mathcal{D}$ *be an unconditional monotonicity term. Then, we can construct multisets* $\mathcal{Z}', \mathcal{D}', \mathcal{M}', \mathcal{S}'$ *such that the following hold:*

(a) $(\mathcal{Z}', \mathcal{D}')$ *are parameters of another integral Shannon-flow inequality witnessed by* $(\mathcal{M}', \mathcal{S}')$*,*
(b) $\mathcal{Z}' \subseteq \mathcal{Z}$ *and* $\mathcal{D}' \subseteq \mathcal{D} - \{\boldsymbol{W}\}$*, with* $|\mathcal{Z}'| \geq |\mathcal{Z}| - 1$*.*
(c) $|\mathcal{D}'| + |\mathcal{M}'| + 2|\mathcal{S}'| \leq |\mathcal{D}| + |\mathcal{M}| + 2|\mathcal{S}| - 1$*.*

In English, the Reset lemma allows us to drop one term $\boldsymbol{W}$ from the RHS of Eq. (6) and obtain a new inequality, by dropping at most one term from the LHS. For a simple illustration, consider the inequality $h(ABC) + h(BCD) \leq h(AB) + h(BC) + h(CD)$ (which we prove in Section 3.1). Suppose we want to drop $h(AB)$: to keep the inequality valid it suffices to drop $h(ABC)$ from the LHS, and obtain $h(BCD) \leq h(BC) + h(CD)$. Suppose instead we want drop $h(BC)$: then we can drop either $h(ABC)$ or $h(BCD)$ from the LHS and the inequality remains valid.

2.2 Disjunctive datalog rules

Let $\boldsymbol{V}$ be a set of variables. An *atom* is an expression of the form $R(\boldsymbol{X})$, where R is a relation symbol, and $\boldsymbol{X} \subseteq \boldsymbol{V}$ is a set of variables. A *schema* Σ is a set of atoms.

Given a finite domain Dom and variables $\boldsymbol{X} \subseteq \boldsymbol{V}$, let $\mathsf{Dom}^{\boldsymbol{X}}$ denote the set of all tuples over the variables in $\boldsymbol{X}$ with values from Dom. Given a tuple $\boldsymbol{t} \in \mathsf{Dom}^{\boldsymbol{V}}$, we use $\boldsymbol{t}_{\boldsymbol{X}}$ to denote the projection of $\boldsymbol{t}$ onto the variables in $\boldsymbol{X}$. A *database instance* D over schema Σ is a mapping that assigns each atom $R(\boldsymbol{X}) \in \Sigma$ to a finite relation $R^D \subseteq \mathsf{Dom}^{\boldsymbol{X}}$. Usually the database instance is clear from the context, so we simply write R instead of R^D. Given a relation $R \subseteq \mathsf{Dom}^{\boldsymbol{X}}$, we use $\mathsf{vars}(R)$ to denote the set of variables $\boldsymbol{X}$.

Given a database instance over schema Σ, the *full natural join* of the instance is the set of tuples $\boldsymbol{t} \in \mathsf{Dom}^{\boldsymbol{V}}$ that satisfy all atoms in Σ:

$$\bowtie\Sigma \stackrel{\text{def}}{=} \{\boldsymbol{t} \in \mathsf{Dom}^{\boldsymbol{V}} \mid \boldsymbol{t}_{\boldsymbol{X}} \in R, \text{ for all } R(\boldsymbol{X}) \in \Sigma\} \tag{7}$$

Definition 2.6 (DDR and its model)**.** Given two schemas Σ_{in} and Σ_{out}, a *disjunctive datalog rule* (DDR) is the expression

$$\bigvee_{Q(\boldsymbol{Z})\in\Sigma_{\mathsf{out}}} Q(\boldsymbol{Z}) \text{ :- } \bigwedge_{R(\boldsymbol{X})\in\Sigma_{\mathsf{in}}} R(\boldsymbol{X}) \tag{8}$$

Given a database instance over the input schema Σ_{in}, a *model* (or *feasible output*) of the DDR (8) is a database instance over the output schema Σ_{out} such that for every tuple $\boldsymbol{t} \in \bowtie \Sigma_{\text{in}}$, there exists at least one atom $Q(\boldsymbol{Z}) \in \Sigma_{\text{out}}$ such that $\boldsymbol{t}_{\boldsymbol{Z}} \in Q(\boldsymbol{Z})$.

Without loss of generality, we assume that if $Q(\boldsymbol{Z}_1), Q(\boldsymbol{Z}_2) \in \Sigma_{\text{out}}$ then $\boldsymbol{Z}_1 \neq \boldsymbol{Z}_2$. We will also write $\boldsymbol{Z} \in \Sigma_{\text{out}}$ to mean that $Q(\boldsymbol{Z}) \in \Sigma_{\text{out}}$ for some relation symbol Q. The *size* of an output instance to a DDR is the total number of tuples in all its relations:

$$\|\Sigma_{\text{out}}\| \stackrel{\text{def}}{=} \sum_{Q(\boldsymbol{Z}) \in \Sigma_{\text{out}}} |Q|. \tag{9}$$

A model of a DDR is *minimal* if no proper subset of it is also a model. Note that a *conjunctive query* (CQ) is a special case of DDR where the output schema Σ_{out} contains only one atom. The answer to a CQ is its unique minimal model.

2.3 Degree constraints and output size bounds for DDRs

Consider a database instance over schema Σ. Let $R \in \Sigma$ be a relation in this instance, and $\delta = (\boldsymbol{Y}|\boldsymbol{X})$ be a monotonicity term. Let $\boldsymbol{x} \in \text{Dom}^{\boldsymbol{X}}$ be a data tuple. Let $R_{|\boldsymbol{X}\cup\boldsymbol{Y}} \stackrel{\text{def}}{=} \text{Dom}^{\boldsymbol{X}\cup\boldsymbol{Y}} \ltimes R$,[1] denote the *restriction* of R onto $\boldsymbol{X} \cup \boldsymbol{Y}$; this restriction is needed to define the degree notions below even in the case when $\boldsymbol{X} \cup \boldsymbol{Y} \not\subseteq \text{vars}(R)$. Note that $R_{|\boldsymbol{X}\cup\boldsymbol{Y}} = \pi_{\boldsymbol{X}\cup\boldsymbol{Y}} R$ when $\boldsymbol{X} \cup \boldsymbol{Y} \subseteq \text{vars}(R)$.

Given a database instance over schema Σ, define the following three quantities:

$$\deg_R(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x}) \stackrel{\text{def}}{=} |\{\boldsymbol{y} \in \text{Dom}^{\boldsymbol{Y}} \mid (\boldsymbol{x}, \boldsymbol{y}) \in R_{|\boldsymbol{X}\cup\boldsymbol{Y}}\}| \tag{10}$$

$$\deg_R(\delta) \stackrel{\text{def}}{=} \max_{\boldsymbol{x} \in \text{Dom}^{\boldsymbol{X}}} \deg_R(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x}) \tag{11}$$

$$\deg_\Sigma(\delta) \stackrel{\text{def}}{=} \min_{R \in \Sigma} \deg_R(\delta). \tag{12}$$

Given a database instance over schema Σ, if N_δ is an integer for which $\deg_\Sigma(\delta) \leq N_\delta$, then we say that the schema Σ satisfies the *degree constraint* (δ, N_δ). More generally, let $\Delta \subseteq \text{MON}$ be a set of monotonicity terms and $\boldsymbol{N} = (N_\delta)_{\delta\in\Delta}$ be a vector of positive integers, one for each monotonicity term in Δ. Then, $(\Delta, \boldsymbol{N})$ is called a set of degree constraints, and we say that the instance Σ *satisfies* $(\Delta, \boldsymbol{N})$ if it satisfies every constraint in it, denoted by $\Sigma \models (\Delta, \boldsymbol{N})$.

The following upper bound from [5, 6] is a generalization of the AGM bound [9]:

Theorem 2.7 ([5, 6]). *Consider a DDR of the form* (8) *with input schema* Σ_{in}*, and output schema* Σ_{out} *such that* $\Sigma_{\text{in}} \models (\Delta, \boldsymbol{N})$*. Assume that there exist two* non-negative *rational weight vectors* $\boldsymbol{w} \stackrel{def}{=} (w_\delta)_{\delta\in\Delta}$ *and* $\boldsymbol{\lambda} \stackrel{def}{=} (\lambda_{\boldsymbol{Z}})_{\boldsymbol{Z}\in\Sigma_{\text{out}}}$ *with* $\|\boldsymbol{\lambda}\|_1 = 1$*, where the following is a Shannon-flow inequality:*

$$\sum_{\boldsymbol{Z}\in\Sigma_{\text{out}}} \lambda_{\boldsymbol{Z}} \cdot h(\boldsymbol{Z}) \leq \sum_{\delta\in\Delta} w_\delta \cdot h(\delta). \tag{13}$$

Then, for any input instance Σ_{in} *of the DDR* (8)*, there exists a model* Σ_{out} *for the DDR for which:*

$$\max_{\boldsymbol{Z}\in\Sigma_{\text{out}}} |Q(\boldsymbol{Z})| \leq \prod_{\delta\in\Delta} N_\delta^{w_\delta} \tag{14}$$

3 Motivating and Running Examples

This section presents two examples to illustrate axis-parallel partitioning and the need for arbitrary hyperplane partitions. The second example is also the running example for the rest of the paper.

[1] $S \ltimes T$ denotes the *semi-join reduce* operator defined by $S \ltimes T \stackrel{\text{def}}{=} \pi_{\text{vars}(S)}(S \bowtie T)$.

3.1 Axis-Parallel Partition

Consider the following DDR (Definition 2.6):

$$U(A,B,C) \vee V(B,C,D) \text{ :- } R(A,B) \wedge S(B,C) \wedge T(C,D) \tag{15}$$

The rule computes two target relations U, and V such that, for any tuple (a,b,c,d) satisfying the body, either $(a,b,c) \in U$ or $(b,c,d) \in V$. In other words, we need to "partition" the output data into two parts, called U and V. The semantics is non-deterministic, and the goal is to compute an output that minimizes $\max(|U|,|V|)$.

Assume all three input relations have size $\leq N$, we show that there exists an output (U,V) such that $|U|,|V| \leq N^{3/2}$. The worst-case optimal algorithm for answering the DDR (15) first partitions S into the "light" and "heavy" parts, based on how large the degree of a value b with respect to C is:

$$S^{\ell}(B,C) := \{(b,c) \in S \mid \deg_S(C|B=b) \leq N^{1/2}\}, \quad S^{h}(B) := \{b \mid \deg_S(C|B=b) > N^{1/2}\}. \tag{16}$$

Then, the algorithm computes U and V both of size $\leq N^{3/2}$ as follows:

$$U(A,B,C) = R(A,B) \wedge S^{\ell}(B,C) \qquad V(B,C,D) = S^{h}(B) \wedge T(C,D) \tag{17}$$

The execution plan consisting of the steps (16)-(17) is derived from a logical query plan based on a proof of the following Shannon-flow inequality:

$$h(ABC) + h(BCD) \leq h(AB) + h(BC) + h(CD) \tag{18}$$

To prove this inequality, it suffices to observe that it can be written as a sum of two simpler inequalities, which hold due to submodularity[2]:

$$h(ABC) \leq h(AB) + h(C|B) \qquad h(BCD) \leq h(B) + h(CD) \tag{19}$$

Next, we describe the main intuition used to derive the execution plan (16)– (17) from the inequalities (19). Intuitively, assume that the polymatroid h is the entropy of some probability space over output tuples (a,b,c,d) of the query. In that case, $h(AB) \leq \log|R|$ and, taking the base of the logarithm to be N to normalize the unit of measurement, we obtain $h(AB), h(BC), h(CD) \leq 1$. The RHS of inequality (18) is $h(AB) + h(BC) + h(CD) \leq 3$, which means that one of the two inequalities (19) must have the RHS $\leq 3/2$. This leads us to the following axis-parallel partitioning of the space of polymatroids:

- If $h(B) \geq 1/2$, then $h(C|B) = h(BC) - h(B) \leq 1/2$ and thus $h(ABC) \leq h(AB) + h(C|B) \leq 3/2$. In the space of degrees, this corresponds to $\deg_S(C|B) \leq N^{1/2}$, and we perform the first join in (17).
- If $h(B) < 1/2$, then $h(BCD) \leq h(B) + h(CD) < 3/2$. This corresponds to $\deg_S(C|B) > N^{1/2}$ and we perform the second join in (17).

The space of polymatroids h is a subset of a 16-dimensional space $\mathbb{R}_+^{2^4}$, and the two cases above partition this space using an axis-parallel hyperplane $h(B) = 1/2$.

3.2 The Hexagon Query and General Hyperplane Partitioning

Now, we present a different example where axis-parallel partitioning is insufficient. Consider the following DDR, due to [38]:

$$Q(A,B,C,D,E,F) \text{ :- } R(A,B,C) \wedge S(C,D,E) \wedge T(E,F,A) \wedge K(B,D,F) \tag{20}$$

The DDR is just a standard CQ, which we call the *hexagon query*. The query is not challenging: if input relations have size $\leq N$, the query can be computed using any worst-case optimal join

[2]By submodularity, $h(C|B) \geq h(C|AB)$ and the first inequality follows from $h(AB) + h(C|B) \geq h(AB) + h(C|AB) = h(ABC)$, while the second is already in the form of a submodularity inequality.

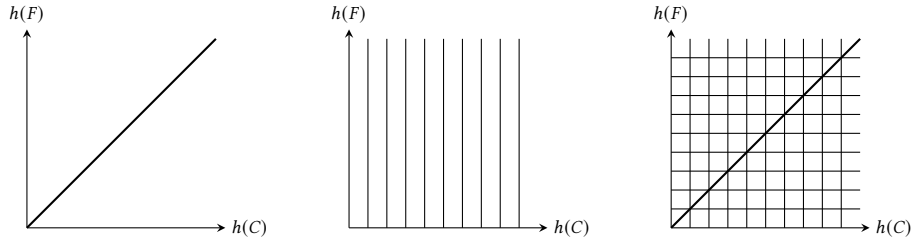


Fig. 1. Space partition for the hexagon query in Eq. (20): While the *hyperplane partition* depicted in (a) is sufficient, PANDA uses $\log^2 N$ *axis-parallel* partitions, depicted in (b) and (c).

(WCOJ) algorithm [32, 33, 37], in time $O\left(N^2 \log N\right)$, which is optimal because the AGM-bound[3] of the query is N^2. However, these WCOJ algorithms do not apply in general to DDRs, and we will not discuss them in this paper. Instead, we want to compute the hexagon query using only data partitioning and joins, in a manner similar to the previous example. The Shannon-flow inequality (Shearer's lemma [15]) that proves $|Q| \le N^2$ is:

$$2h(ABCDEF) \le h(ABC) + h(CDE) + h(EFA) + h(BDF) \tag{21}$$

This inequality is also a sum of two simpler inequalities, which form the basis of the proof sequence in Theorem 2.4:

$$h(ABCDEF) \le h(ABC) + h(DE|C) + h(F) \qquad h(ABCDEF) \le h(EFA) + h(BD|F) + h(C) \tag{22}$$

Assuming that h satisfies $h(ABC), h(CDE), h(EFA), h(BDF) \le 1$, we seek a partition of the space of polymatroids such that, in each part, one of the two RHS in (22) is ≤ 2. However, no axis-parallel partition exists with this property. If it existed, it would correspond to partitioning the values of C and F into light and heavy, but then the first join derived from (22) joins the light-C with heavy-F, while the second joins the heavy-C with the light-F, which leaves out the combinations light-light and heavy-heavy. Clearly, no algorithm is possible by simply partitioning into light and heavy.

Instead, the natural partition of the polymatroids uses the hyperplane $h(C) = h(F)$, as follows:

- If $h(C) \ge h(F)$ then $h(ABCDEF) \le h(ABC) + h(DE|C) + h(F) \le h(ABC) + h(DE|C) + h(C) = h(ABC) + h(CDE) \le 2$.
- If $h(C) \le h(F)$ then $h(ABCDEF) \le h(EFA) + h(BD|F) + h(C) \le h(EFA) + h(BD|F) + h(F) = h(EFA) + h(BDF) \le 2$.

The corresponding hyperplane-based space partition is shown in Fig. 1a. In order to compute the query by using only axis-parallel partitions, PANDA uses a poly-logarithmic number of buckets: It partitions the values c into $\log N$ buckets such that in bucket i, $2^{i-1} < \deg_S(DE|C = c) \le 2^i$, as shown in Fig. 1b, then partitions similarly the values f into $\log N$ buckets, as in Fig. 1c. Then, in each of the $\log^2 N$ partitions, PANDA makes a choice whether to use the left or the right inequality in (22). This is an expensive way to choose between only two joins represented by the inequalities (22), and this is the reason why PANDA has an extra $\log^2 N$ factor for this query. In contrast, we will show later that our PANDAExpress algorithm uses only the hyperplane $h(C) = h(F)$ to partition this instance, thus removing the $\log^2 N$ overhead.

[3]See the Appendix of https://arxiv.org/abs/2512.10217 for more challenging examples.

4 An Inequality on Sub-Probability Measures

This section presents our first main technical contribution: a new inequality over *sub-probability measures*, which is a concept that we review next. This inequality forms the basis of a new algorithm PANDAExpress, and it is the probabilistic analogue of the Shannon-flow inequality (13) that underpins PANDA.

4.1 Sub-probability measures

We review some very basic facts from finite and discrete measure theory [36]. All our measures are finite and discrete in this paper, so we will drop the adjectives "finite and discrete" from now on. Let Ω be a finite set. A function $p : 2^\Omega \to R_+$ is called a *measure* on Ω if $p(\emptyset) = 0$ and $p(A) = \sum_{x\in A} p(x)$ for any $A \subseteq \Omega$. In particular, to define a measure on Ω, it is sufficient to define the values of $p(x)$ for all $x \in \Omega$. It is called a *sub-probability measure* if $p(\Omega) \leq 1$, and a *probability measure* if $p(\Omega) = 1$.

Let p be a measure on Ω. If $\Omega = \Omega_1 \times \Omega_2$ is a Cartesian product of two domains, then the measure $p_1(x_1) \stackrel{\text{def}}{=} \sum_{x_2\in\Omega_2} p(x_1, x_2), \forall x_1 \in \Omega_1$ is called the *marginal measure* of p on Ω_1. Let $A \subseteq \Omega$ with positive measure $p(A) > 0$, then the measure $p_{|A}(B) \stackrel{\text{def}}{=} \frac{p(B\cap A)}{p(A)}$ for any $B \subseteq \Omega$ is called the *conditional measure* of p on A. Typically we write $p(B|A)$ instead of $p_{|A}(B)$. It is easy to see that, if p is a sub-probability measure then any of its marginals is also a sub-probability measure, and any of its conditional measures is a probability measure.

In this paper, we will deal with domains $\Omega = \text{Dom}^{\boldsymbol{X}}$ where $\boldsymbol{X}$ is some set of variables, and Dom is a finite set. A sub-probability measure p on $\text{Dom}^{\boldsymbol{X}}$ will be denoted by $p_{\boldsymbol{X}}$. The aforementioned facts are specialized as follows.

Proposition 4.1. *Given a sub-probability measure $p_{\boldsymbol{XY}}$, the following hold:*

- *The following marginal measure is a sub-probability measure:*

$$p_{\boldsymbol{X}}(\boldsymbol{x}) \stackrel{def}{=} \sum_{\boldsymbol{y}\in Dom^{\boldsymbol{Y}}} p_{\boldsymbol{XY}}(\boldsymbol{x}, \boldsymbol{y}) \tag{23}$$

- *Given $\boldsymbol{x} \in Dom^{\boldsymbol{X}}$ the following conditional measure is a sub-probability measure:*

$$p_{\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}}(\boldsymbol{y}) \stackrel{def}{=} \begin{cases} \frac{p_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y})}{p_{\boldsymbol{X}}(\boldsymbol{x})} & \text{if } p_{\boldsymbol{X}}(\boldsymbol{x}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

As is customary in probability theory, we write $p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$ instead of $p_{\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}}(\boldsymbol{y})$, with the implicit understanding that there is a measure for each $\boldsymbol{x} \in \text{Dom}^{\boldsymbol{X}}$.

Definition 4.2. Given k sub-probability measures $p_1, p_2, \ldots, p_k$ on the same domain Ω, their *geometric mean* is a sub-probability measure p defined by $p(x) \stackrel{\text{def}}{=} \left(\prod_{i=1}^k p_i(x)\right)^{\frac{1}{k}}, \forall x \in \Omega$.

Proposition 4.3. *The geometric mean of sub-probability measures is also a sub-probability measure.*

Proof. This follows trivially from the AM-GM inequality:

$$p(\Omega) = \sum_{x\in\Omega} \left(\prod_{i=1}^k p_i(x)\right)^{\frac{1}{k}} \leq \sum_{x\in\Omega} \frac{1}{k} \sum_{i=1}^k p_i(x) = \frac{1}{k} \sum_{i=1}^k \sum_{x\in\Omega} p_i(x) \leq 1$$

□

4.2 The New Inequality

Given a monotonicity term $\delta = (Y|X)$, a tuple $\boldsymbol{t} \in \mathrm{Dom}^{\boldsymbol{V}}$ where $\boldsymbol{V} \supseteq \boldsymbol{X} \cup \boldsymbol{Y}$, and a conditional sub-probability measure $p_{Y|X}$, define the following shorthand notation:

$$p_\delta(\boldsymbol{t}) \stackrel{\text{def}}{=} p_{Y|X}(\boldsymbol{t}_Y|\boldsymbol{t}_X) \tag{25}$$

In particular, if $\delta = (\boldsymbol{Z})$ was an unconditional monotonicity term, then $p_{\boldsymbol{Z}}(\boldsymbol{t}) \stackrel{\text{def}}{=} p_{\boldsymbol{Z}}(\boldsymbol{t}_{\boldsymbol{Z}})$. We prove the following inequality.

Lemma 4.4. *Let $\boldsymbol{w} \stackrel{def}{=} (w_\delta)_{\delta\in\Delta}$ and $\boldsymbol{\lambda} \stackrel{def}{=} (\lambda_{\boldsymbol{Z}})_{\boldsymbol{Z}\in\Sigma_{\text{out}}}$ with $\|\boldsymbol{\lambda}\|_1 = 1$, be any two non-negative rational weight vectors for which* (13) *is a Shannon-flow inequality over $\boldsymbol{V}$. Then, given any collection of sub-probability measures p_δ, $\delta \in \Delta$, there exists a collection of sub-probability measures $p_{\boldsymbol{Z}}$, $\boldsymbol{Z} \in \Sigma_{\text{out}}$, such that*

$$\prod_{\boldsymbol{Z}\in\Sigma_{\text{out}}} p_{\boldsymbol{Z}}(\boldsymbol{t})^{\lambda_{\boldsymbol{Z}}} \geq \prod_{\delta\in\Delta} p_\delta(\boldsymbol{t})^{w_\delta}, \qquad \forall \boldsymbol{t} \in Dom^{\boldsymbol{V}} \tag{26}$$

Proof. Write $\lambda_{\boldsymbol{Z}} = \frac{m_{\boldsymbol{Z}}}{d}$ and $w_\delta = \frac{m_\delta}{d}$ where $m_{\boldsymbol{Z}}$ and m_δ are positive integers, and d is the minimum common denominator of the coefficients. By multiplying both sides of (13) by d, we obtain the integral Shannon-flow inequality (6): $\sum_{\boldsymbol{Z}\in\mathcal{Z}} h(\boldsymbol{Z}) \leq \sum_{\delta\in\mathcal{D}} h(\delta)$ in which $\mathcal{Z}$ is a multiset of elements in Σ_{out} and $\mathcal{D}$ is a multiset of elements in Δ. Furthermore, elements $\boldsymbol{Z} \in \mathcal{Z}$ appear with multiplicity $m_{\boldsymbol{Z}}$, and elements $\delta \in \mathcal{D}$ appear with multiplicity m_δ.

We first claim that there exist sub-probability measures $p_{\boldsymbol{Z}}$, one for each copy of $\boldsymbol{Z}$ in $\mathcal{Z}$ for which

$$\prod_{\boldsymbol{Z}\in\mathcal{Z}} p_{\boldsymbol{Z}}(\boldsymbol{t}) \geq \prod_{\delta\in\mathcal{D}} p_\delta(\boldsymbol{t}), \qquad \forall \boldsymbol{t} \in \mathrm{Dom}^{\boldsymbol{V}} \tag{27}$$

To prove this claim, our strategy is to examine the proof sequence of the integral Shannon-flow inequality from Theorem 2.4, where each step turns the multiset $\mathcal{D}$ into a multiset $\mathcal{D}'$ by applying one of the four proof steps. It is sufficient to show that, for each proof step $\mathcal{D} \to \mathcal{D}'$, we can construct sub-probability measures p_δ for all $\delta \in \mathcal{D}'$ such that $\prod_{\delta\in\mathcal{D}} p_\delta(\boldsymbol{t}) \leq \prod_{\delta\in\mathcal{D}'} p_\delta(\boldsymbol{t})$, because at the last step we will obtain $\mathcal{D}' \supseteq \mathcal{Z}$, which implies $\prod_{\delta\in\mathcal{D}'} p_\delta(\boldsymbol{t}) \leq \prod_{\boldsymbol{Z}\in\mathcal{Z}} p_{\boldsymbol{Z}}(\boldsymbol{t})$. This is done via trivial case analysis on the four proof steps.

- *Monotonicity:* $h(\boldsymbol{XY}) \to h(\boldsymbol{X})$. Here, remove $\boldsymbol{XY}$ and add $\boldsymbol{X}$ to $\mathcal{D}$ to obtain $\mathcal{D}'$. Correspondingly, we replace $p_{\boldsymbol{XY}}$ by its marginal $p_{\boldsymbol{X}}$ defined in (23), where $p_{\boldsymbol{XY}}(\boldsymbol{t}) \leq p_{\boldsymbol{X}}(\boldsymbol{t})$ is trivial to verify.
- *Submodularity:* $h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{Y}|\boldsymbol{XZ})$. In this case, replace $p_{\boldsymbol{Y}|\boldsymbol{X}}$ by the conditional measure $p_{\boldsymbol{Y}|\boldsymbol{XZ}}$, defined by $p_{\boldsymbol{Y}|\boldsymbol{XZ}}(\boldsymbol{y}|\boldsymbol{xz}) \stackrel{\text{def}}{=} p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$. This way, we have not changed the product at all.
- *Composition:* $h(\boldsymbol{X})+h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{XY})$. Here, remove $\boldsymbol{X}$ and $\boldsymbol{Y}|\boldsymbol{X}$ from $\mathcal{D}$ and add $\boldsymbol{XY}$ to obtain $\mathcal{D}'$. Correspondingly, we keep the product unchanged by defining $p_{\boldsymbol{XY}}(\boldsymbol{x},\boldsymbol{y}) \stackrel{\text{def}}{=} p_{\boldsymbol{X}}(\boldsymbol{x}) \cdot p_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{x})$.
- *Decomposition:* $h(\boldsymbol{XY}) \to h(\boldsymbol{X})+h(\boldsymbol{Y}|\boldsymbol{X})$. Here, remove $\boldsymbol{XY}$ from $\mathcal{D}$ and add $\boldsymbol{X}$ and $\boldsymbol{Y}|\boldsymbol{X}$ to obtain $\mathcal{D}'$. Correspondingly, we keep the product unchanged by defining the marginal and conditional measures exactly as shown in (23) and (24).

The claim is thus proved. Next, note that the RHS of (27) is

$$\prod_{\delta\in\mathcal{D}} p_\delta(\boldsymbol{t}) = \prod_{\delta\in\Delta} p_\delta(\boldsymbol{t})^{m(\delta)} = \prod_{\delta\in\Delta} p_\delta(\boldsymbol{t})^{d\cdot w_\delta} = \left(\prod_{\delta\in\Delta} p_\delta(\boldsymbol{t})^{w_\delta}\right)^d \tag{28}$$

Hence, it remains to show that there exists a set of sub-probability measures p_Z for all $Z \in \Sigma_{\text{out}}$ for which

$$\prod_{Z \in \Sigma_{\text{out}}} p_Z(\boldsymbol{t})^{m(Z)} = \prod_{Z \in \mathcal{Z}} p_Z(\boldsymbol{t}), \qquad \forall \boldsymbol{t} \in \text{Dom}^V \tag{29}$$

To see this, for each $Z \in \Sigma_{\text{out}}$, set p_Z to be the geometric mean of m_Z sub-probability measures p_Z for copies of Z in $\mathcal{Z}$, then we obtain equality (29). The proof is complete because by Proposition 4.3 the geometric mean of sub-probability measures is also a sub-probability measure. □

Lemma 4.4 gives us another way to prove the output size bound of DDR in Theorem 2.7, and this proof will directly lead to the PANDAExpress algorithm described in the next section.

Corollary 4.5. *Theorem 2.7 holds.*

Proof. For each $\delta = (Y|X) \in \Delta$, let $R \in \Sigma_{\text{in}}$ be a relation where $\max_{\boldsymbol{x}} \deg_R(Y|X = \boldsymbol{x}) \le N_\delta$. Define the following conditional sub-probability measure:

$$p_\delta(\boldsymbol{y}|\boldsymbol{x}) \stackrel{\text{def}}{=} \begin{cases} \frac{1}{N_\delta} & (\boldsymbol{x}, \boldsymbol{y}) \in R_{|X \cup Y} \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

Write $B \stackrel{\text{def}}{=} \prod_{\delta \in \Delta} N_\delta^{w_\delta}$. From Lemma 4.4, there exists a sub-probability measure p_Z for each $Z \in \Sigma_{\text{out}}$ such that

$$\prod_{Z \in \Sigma_{\text{out}}} p_Z(\boldsymbol{t})^{\lambda_Z} \ge \prod_{\delta \in \Delta} p_\delta(\boldsymbol{t})^{w_\delta} = \prod_{\delta \in \Delta} \left(\frac{1}{N_\delta}\right)^{w_\delta} = \frac{1}{B}, \qquad \forall \boldsymbol{t} \in \bowtie \Sigma_{\text{in}} \tag{31}$$

To show (14), it is sufficient to show that there exists a model Σ_{out} for the DDR (8) such that $|Q(Z)| \le B$ for all $Q(Z) \in \Sigma_{\text{out}}$. This model is constructed by defining

$$Q(Z) \stackrel{\text{def}}{=} \left\{ \boldsymbol{z} \in \text{Dom}^Z \mid p_Z(\boldsymbol{z}) \ge \frac{1}{B} \right\} \qquad \forall Z \in \Sigma_{\text{out}} \tag{32}$$

Obviously, $|Q(Z)| \le B$ because p_Z is a sub-probability measure. To show that this is indeed a model for the DDR (8), note that since $\|\boldsymbol{\lambda}\|_1 = 1$, for every $\boldsymbol{t} \in \bowtie \Sigma_{\text{in}}$ inequality (31) implies that there exists some $Z \in \Sigma_{\text{out}}$ such that $p_Z(\boldsymbol{t}) \ge \frac{1}{B}$, which means $\boldsymbol{t}_Z \in Q(Z)$. □

4.3 Illustration of the Inequality on the Hexagon Query

To illustrate why inequality (26) holds and how it can be used to prove the output size bound of DDRs, this section presents an alternative proof of the size bound $|Q| \le N^2$ for the hexagon query (20). Recall that the typical proof of this fact is to make use of the AGM bound which is the direct consequence of Shearer's lemma [15]. For this query, Shearer's lemma is *exactly* the Shannon-flow inequality (21).

We start from the inequality (21) and write down the *proof sequence* (Theorem 2.4) that transforms the RHS into the LHS. This proof sequence is depicted in Fig. 2 (left). Now, to each term on the RHS of (21), we associate a probability measure. In particular, to $h(ABC)$, we associate a measure p_{ABC} defined as $p_{ABC}(abc) \stackrel{\text{def}}{=} 1/N$ if the tuple (a, b, c) is in R, and 0 otherwise. Similarly, we define measures p_{CDE}, p_{EFA}, and p_{BDF} for S, T, and K respectively. By construction, we have:

$$p_{ABC}(abc) \cdot p_{CDE}(cde) \cdot p_{EFA}(efa) \cdot p_{BDF}(bdf) = 1/N^4, \qquad \forall (a, b, c, d, e, f) \in Q. \tag{33}$$

Then, we trace the proof steps in Fig. 2: Each proof step replaces (one or two) entropy terms by new entropy terms. We associate to each new entropy term a new probability measure, thus mirroring the proof step in the measure world.

- From the first step $h(CDE) \to h(C) + h(DE|C)$ we take the measure p_{CDE} (associated to $h(CDE)$) and define two new measures p_C and $p_{DE|C}$ as its *marginal* and *conditional* (Eq. (23) and (24)), and associate them to $h(C)$ and $h(DE|C)$ respectively. Note that by definition, $p_{CDE} = p_C \cdot p_{DE|C}$, hence if we replace $p_{CDE}(cde)$ with $p_C(c) \cdot p_{DE|C}(de|c)$ in Eq. (33), the equality still holds.
- Similarly, the second step $h(ABC) + h(DE|C) \to h(ABCDE)$ tells us to take p_{ABC} and $p_{DE|C}$, define measure p_{ABCDE} as their product, and associate it to $h(ABCDE)$. To track this move, replace the product $p_{ABC}(abc) \cdot p_{DE|C}(de|c)$ in the new Eq. (33) with $p_{ABCDE}(abcde)$.
- Following this process until the end of the proof sequence, we end up with two different measures $p'(ABCDEF), p''(ABCDEF)$ corresponding to $2h(ABCDEF)$ on the LHS of Eq. (21). At the same time, Eq. (33) becomes:

$$p'_{ABCDEF}(abcdef) \cdot p''_{ABCDEF}(abcdef) = 1/N^4, \qquad \forall (a,b,c,d,e,f) \in Q. \tag{34}$$

- Finally, define a new measure p_{ABCDEF} as the *geometric mean* of p'_{ABCDEF} and p''_{ABCDEF} (Proposition 4.3). Hence, p_{ABCDEF} takes a value of $1/N^2$ for all tuples $(a,b,c,d,e,f) \in Q$. This implies that the size of the support of p_{ABCDEF}, which is exactly $|Q|$, is at most N^2, as desired.

Proof Steps	Sub-probability Measures
$h(CDE) \to h(C) + h(DE\|C)$	$p_C(c) \stackrel{\text{def}}{=} \sum_{de} p_{CDE}(cde) = \dfrac{\deg_S(DE\|C=c)}{N}$ $p_{DE\|C}(de\|c) \stackrel{\text{def}}{=} \dfrac{p_{CDE}(cde)}{p_C(c)} = \dfrac{1}{\deg_S(DE\|C=c)}$
$h(ABC) + h(DE\|C) \to h(ABCDE)$	$p_{ABCDE}(abcde) \stackrel{\text{def}}{=} p_{ABC}(abc) \cdot p_{DE\|C}(de\|c) = \dfrac{1}{N \cdot \deg_S(DE\|C=c)}$
$h(BDF) \to h(F) + h(BD\|F)$	$p_F(f) \stackrel{\text{def}}{=} \sum_{bd} p_{BDF}(bdf) = \dfrac{\deg_K(BD\|F=f)}{N}$ $p_{BD\|F}(bd\|f) \stackrel{\text{def}}{=} \dfrac{p_{BDF}(bdf)}{p_F(f)} = \dfrac{1}{\deg_K(BD\|F=f)}$
$h(EFA) + h(BD\|F) \to h(ABDEF)$	$p_{ABDEF}(abdef) \stackrel{\text{def}}{=} p_{EFA}(efa) \cdot p_{BD\|F}(bd\|f) = \dfrac{1}{N \cdot \deg_K(BD\|F=f)}$
$h(ABCDE) + h(F) \to h(ABCDEF)$	$p'_{ABCDEF}(abcdef) \stackrel{\text{def}}{=} p_{ABCDE}(abcde) \cdot p_F(f) = \dfrac{\deg_K(BD\|F=f)}{N^2 \cdot \deg_S(DE\|C=c)}$
$h(ABDEF) + h(C) \to h(ABCDEF)$	$p''_{ABCDEF}(abcdef) \stackrel{\text{def}}{=} p_{ABDEF}(abdef) \cdot p_C(c) = \dfrac{\deg_S(DE\|C=c)}{N^2 \cdot \deg_K(BD\|F=f)}$

Fig. 2. A new proof of the bound $|Q| \le N^2$ for the hexagon query Q in Eq. (20), using sub-probability measures. The depicted proof sequence is for the Shannon-flow inequality (21) and it is a slightly more compacted version of the proof sequence obtained from Theorem 2.4, where we merge, for example, two steps $h(DE|C) \to h(DE|ABC)$ and $h(ABC) + h(DE|ABC) \to h(ABCDE)$ into one $h(ABC) + h(DE|C) \to h(ABCDE)$. For simplicity, we assume above that all divisions by zero produce zero (similar to Eq. (24)).

5 The PANDAExpress algorithm

5.1 Illustration of PANDAExpress on the Hexagon Query

This section explains intuitively how the size bound proof of the Hexagon query shown in Section 4.3 can be turned into an actual algorithm. The formal presentation of the PANDAExpress algorithm is given in the next section.

The proof shown in Section 4.3 can *almost* be turned directly into an $O(N^2)$-algorithm for computing the hexagon query. All we need to do is to actually *materialize* all the intermediate sub-probability measures defined in the proof. Of course, we only need to materialize the non-zero values of each measure.

However, there is an issue we have to deal with: The measures p'_{ABCDEF} and p''_{ABCDEF} defined before could potentially have supports larger than N^2, exceeding the runtime budget of $O(N^2)$. The minor twist in our algorithm is to observe that we do not need to compute the full measures p'_{ABCDEF} and p''_{ABCDEF}. Instead of computing them fully, we will only compute *their truncations*, $\overline{p}'_{ABCDEF}$ and $\overline{p}''_{ABCDEF}$ respectively, where $\overline{p}'_{ABCDEF}$ results from p'_{ABCDEF} by only keeping the values that are $\geq 1/N^2$, and similarly for $\overline{p}''_{ABCDEF}$. By construction, the supports of $\overline{p}'_{ABCDEF}$ and $\overline{p}''_{ABCDEF}$ are no larger than N^2. Moreover, by Eq. (34), for every tuple $(a, b, c, d, e, f) \in Q$, either $p'_{ABCDEF}(abcdef) \geq 1/N^2$ or $p''_{ABCDEF}(abcdef) \geq 1/N^2$ (or both), hence the tuple (a, b, c, d, e, f) will occur in the support of at least one of $\overline{p}'_{ABCDEF}$ or $\overline{p}''_{ABCDEF}$. After computing $\overline{p}'_{ABCDEF}$ and $\overline{p}''_{ABCDEF}$, we return the union of their supports as the output of the query.

So where does the partitioning hyperplane $h(C) = h(F)$ mentioned in Section 3.2 come into play in this algorithm? Consider the definitions of $p'_{ABCDEF}(abcdef)$ and $p''_{ABCDEF}(abcdef)$ in Fig. 2. Note that the condition $p'_{ABCDEF}(abcdef) \geq 1/N^2$ corresponds to the following inequality:

$$\deg_K(BD|F = f) \geq \deg_S(DE|C = c), \tag{35}$$

Similarly, the condition $p''_{ABCDEF}(abcdef) \geq 1/N^2$ corresponds to the opposite inequality. Hence, in the degree space, the above algorithm is partitioning the output into two parts, using the boundary $\deg_K(BD|F = f) = \deg_S(DE|C = c)$. This condition *is* the interpretation of the hyperplane $h(C) = h(F)$ in the degree space, thus completing the connection.

In summary, proof sequences form the basis of a *logical query plan*, whereas the sub-probability measures form the basis for the actual *execution plan* to answer the query.

5.2 The Algorithm

We are now ready to formally present our PANDAExpress algorithm, given in Algorithm 1. We shall prove its correctness and analyze its runtime in the next section. The algorithm takes as input three parameters: $\mathcal{Z}$ is a multiset of unconditional monotonicity terms, $\mathcal{D}$ is a multiset of monotonicity terms, and $\mathcal{P}$ is a set of sub-probability measures, one for each copy of monotonicity term in $\mathcal{D}$. The pair $(\mathcal{Z}, \mathcal{D})$ are parameters of an integral Shannon-flow inequality of the form (6) witnessed by $(\mathcal{M}, \mathcal{S})$. We think of $\mathcal{P}$ as an *annotated* database instance where each tuple is annotated with a probability. We explain later (Eq. (40)) how to initialize these annotations from the input database instance to a DDR. There is a global "budget" parameter B which is a positive integer that we will define later (Eq. (39)) in the proof of correctness.

The algorithm is recursive. In the base case, if $\mathcal{D}$ contains any $Z \in \mathcal{Z}$, then we can directly construct and return a relation $Q(Z)$ by collecting tuples in the support of the corresponding sub-probability measure p_Z.

If the base case does not hold, then from Theorem 2.4 there exists a proof sequence for Eq. (6) whose first step is s. The sub-routine apply-step modifies $\mathcal{D}$ and $\mathcal{P}$ slightly to obtain a new multiset of monotonicity terms $\mathcal{D}^\ell$ and a new set of sub-probability measures $\mathcal{P}^\ell$. We will explain

Algorithm 1 PANDAExpress$(\mathcal{Z}, \mathcal{D}, \mathcal{P})$

Inputs: $(\mathcal{Z}, \mathcal{D})$ form an integral Shannon-flow inequality Eq. (6)
$\mathcal{P}$ is a set of sub-probability measures, one for each $\delta \in \mathcal{D}$

if $\mathcal{D}$ contains any $Z \in \mathcal{Z}$ **then**
  return $Q(Z) := \{z \mid p_Z(z) > 0\}$ ▷ Tuples in the support of p_Z
end if
Let s be the first step in a proof sequence of the Shannon-flow inequality defined by $(\mathcal{Z}, \mathcal{D})$.
$(\mathcal{D}^\ell, \mathcal{P}^\ell) \leftarrow$ apply-step$(s, \mathcal{D}, \mathcal{P})$
$\Sigma^\ell_{\text{out}} \leftarrow$ PANDAExpress$(\mathcal{Z}, \mathcal{D}^\ell, \mathcal{P}^\ell)$ ▷ Light branch; continue on with the proof sequence
if s is not a composition step **or** $|\mathcal{Z}| = 1$ **then**
  return Σ^ℓ_{out} ▷ No heavy branch
else if s is a composition step $h(X) + h(Y|X) \to h(XY)$ **and** $|\mathcal{Z}| > 1$ **then**
  $(\mathcal{Z}^h, \mathcal{D}^h) \leftarrow$ reset$(\mathcal{Z}, \mathcal{D}^\ell, XY)$ ▷ Apply Reset Lemma 2.5 with $\boldsymbol{W} = \boldsymbol{XY}$
  $\mathcal{P}^h \leftarrow \{p_\delta \mid p_\delta \in \mathcal{P}^\ell \text{ and } \delta \in \mathcal{D}^h\}$ ▷ Note that $\mathcal{D}^h \subseteq \mathcal{D}^\ell - \{XY\}$
  $\Sigma^h_{\text{out}} \leftarrow$ PANDAExpress$(\mathcal{Z}^h, \mathcal{D}^h, \mathcal{P}^h)$ ▷ Heavy branch
  return $\Sigma^\ell_{\text{out}} \cup \Sigma^h_{\text{out}}$
end if

shortly what this modification is. It then recursively calls the algorithm with the new parameters $(\mathcal{Z}, \mathcal{D}^\ell, \mathcal{P}^\ell)$, applying the next step in the proof sequence. This is the "light" branch of the recursion. The result is returned if the step is *not* a composition step, in which case there is no "heavy" branch.

If s is a composition step and $|\mathcal{Z}| > 1$, then we need to do more work, creating a heavy branch. The reset call applies Lemma 2.5 with $\boldsymbol{W} = \boldsymbol{XY}$ to obtain a new Shannon-flow inequality defined by $(\mathcal{Z}^h, \mathcal{D}^h)$, where $\mathcal{Z}^h \subseteq \mathcal{Z}$, $|\mathcal{Z}^h| \geq |\mathcal{Z}| - 1$, and $\mathcal{D}^h \subseteq \mathcal{D}^\ell - \{XY\}$. We then define $\mathcal{P}^h$ by restricting $\mathcal{P}^\ell$ to only those p_δ where $\delta \in \mathcal{D}^h$. With these new parameters, we recursively call the algorithm again to obtain Σ^h_{out}. Finally, the union of the heavy and light branches is returned.

To complete the description of the algorithm, we now explain the sub-routine apply-step, which modifies $\mathcal{D}$ and $\mathcal{P}$ according to the proof step s.

- If s is a *decomposition* step $h(XY) \to h(X) + h(Y|X)$, then let p_X be the marginal measure (23) of p_{XY} on $\boldsymbol{X}$ and $p_{Y|X}$ be the conditional measure (24) of p_{XY} on $\boldsymbol{Y}$ given $\boldsymbol{X}$.

$$\mathcal{D}^\ell \leftarrow \mathcal{D} \setminus \{XY\} \cup \{X, (Y|X)\} \qquad \mathcal{P}^\ell \leftarrow \mathcal{P} \setminus \{p_{XY}\} \cup \{p_X, p_{Y|X}\}$$

- If s is a *submodularity* step $h(Y|X) \to h(Y|XZ)$, then define $p_{Y|XZ} \stackrel{\text{def}}{=} p_{Y|X}$ and

$$\mathcal{D}^\ell \leftarrow \mathcal{D} \setminus \{(Y|X)\} \cup \{(Y|XZ)\} \qquad \mathcal{P}^\ell \leftarrow \mathcal{P} \setminus \{p_{Y|X}\} \cup \{p_{Y|XZ}\}$$

- If s is a *monotonicity* step $h(XY) \to h(X)$, then define p_X as the marginal measure (23) of p_{XY} on $\boldsymbol{X}$, and

$$\mathcal{D}^\ell \leftarrow \mathcal{D} \setminus \{XY\} \cup \{X\} \qquad \mathcal{P}^\ell \leftarrow \mathcal{P} \setminus \{p_{XY}\} \cup \{p_X\}$$

- If s is a *composition* step $h(X) + h(Y|X) \to h(XY)$, then

$$\mathcal{D}^\ell \leftarrow \mathcal{D} \setminus \{X, (Y|X)\} \cup \{XY\} \qquad \mathcal{P}^\ell \leftarrow \mathcal{P} \setminus \{p_X, p_{Y|X}\} \cup \{p_{XY}\}$$

  where p_{XY} is the *truncated* product measure:

$$p_{XY}(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \begin{cases} p_X(\boldsymbol{x}) \cdot p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}), & \text{if } p_X(\boldsymbol{x}) \cdot p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) \geq 1/B \\ 0, & \text{otherwise} \end{cases} \tag{36}$$

6 Proof of Correctness and Runtime Analysis

Theorem 6.4 is the main result of the paper. Before stating and proving it, we need to introduce an auxiliary optimization problem and its properties, which are of independent interest and will be used again in Section 7.

6.1 An Auxiliary Optimization Problem

In order to minimize the upper bound of the output size, as given in Theorem 2.7, which is also the same runtime expression in Theorem 6.4, we want to find coefficients $\boldsymbol{w}, \boldsymbol{\lambda}$ that minimize the right-hand side of (14) subject to the Shannon-flow inequality (13). In particular, we want to solve the optimization problem stated in Proposition 6.3 below. The result was implicit in [5, 6]; however, we include the proof here for completeness, in part because the language of the statement in the proposition is about the primal problem, while the one in [5, 6] is about the dual problem, and hence they look quite different.

Definition 6.1. Let $(\Delta, \boldsymbol{N})$ denote a set of degree constraints over variables $\boldsymbol{V}$. We write $h \models (\Delta, \boldsymbol{N})$ to denote the fact that h is a polymatroid on $\boldsymbol{V}$ that satisfies all degree constraints in $(\Delta, \boldsymbol{N})$, i.e., for every $\delta = (Y|X) \in \Delta$, we have $h(\delta) \stackrel{\text{def}}{=} h(XY) - h(X) \le \log N_\delta$.

Note that, including the constraints that h is a polymatroid, the set of constraints $h \models (\Delta, \boldsymbol{N})$ can be expressed as a set of linear inequalities.

The following lemma shows that the optimization problem in Proposition 6.3 below can be converted into a linear program. We isolate this lemma because we will use it again to convert the primal form of the fractional hypertree width and the submodular width into their dual forms in Section 7. Note that the lemma was basically proved in Lemma 6.4 in [6], but it was stated using the dual form of the optimization problem.

Lemma 6.2 ([6])**.** *Let $(\Delta, \boldsymbol{N})$ be a set of degree constraints over variables $\boldsymbol{V}$. Then, the following optimization problem*

$$\begin{aligned} \min_{\boldsymbol{\lambda}, \boldsymbol{w}} \quad & \sum_{\delta \in \Delta} w_\delta \log N_\delta \\ \text{s.t.} \quad & \sum_{Z \in \Sigma_{\text{out}}} \lambda_Z \cdot h(Z) \le \sum_{\delta \in \Delta} w_\delta \cdot h(\delta) \text{ is a Shannon-flow inequality} \\ & \|\boldsymbol{\lambda}\|_1 = 1 \text{ and } \boldsymbol{\lambda} \ge \boldsymbol{0}, \boldsymbol{w} \ge \boldsymbol{0} \end{aligned}$$

has exactly the same objective value as the following optimization problem, provided that one of them is bounded:

$$\max_{\boldsymbol{h} \models (\Delta, \boldsymbol{N})} \min_{Z \in \Sigma_{\text{out}}} h(Z) \tag{37}$$

Moreover, Problem (37) *can be expressed as a linear program.*

Proof. Let Γ_n denote the set of all polymatroids on n variables, where $n := |\boldsymbol{V}|$. As in the proof of Lemma 6.4 in [6], we reformulate (37) as the following linear program, which proves the last statement of the lemma:

$$\text{opt} \stackrel{\text{def}}{=} \max_{t, \boldsymbol{h} \in \Gamma_n} \{ t \mid t \le h(Z), \forall Z \in \Sigma_{\text{out}}, \text{ and } h(\delta) \le \log N_\delta, \forall \delta \in \Delta \}$$

Let λ_Z and w_δ be the Lagrange multipliers associated with the constraints $t \le h(Z)$ and $h(\delta) \le \log N_\delta$ respectively. The Lagrangian dual function is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{w}) &\stackrel{\text{def}}{=} \max_{t, \boldsymbol{h} \in \Gamma_n} \left\{ t + \sum_{Z \in \Sigma_{\text{out}}} \lambda_Z (h(Z) - t) + \sum_{\delta \in \Delta} w_\delta (\log N_\delta - h(\delta)) \right\} \\ &= \sum_{\delta \in \Delta} w_\delta \log N_\delta + \max_t (1 - \|\boldsymbol{\lambda}\|_1) t + \max_{\boldsymbol{h} \in \Gamma_n} \left\{ \sum_{Z \in \Sigma_{\text{out}}} \lambda_Z h(Z) - \sum_{\delta \in \Delta} w_\delta h(\delta) \right\}\end{aligned}$$

Since the optimization problem is linear, strong duality holds, and thus the Lagrangian dual problem has the same objective as the primal problem, namely $\text{opt} = \min_{\boldsymbol{\lambda}, \boldsymbol{w} \ge \boldsymbol{0}} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{w})$, which we had assumed to be bounded.

If $(\boldsymbol{\lambda}^*, \boldsymbol{w}^*)$ is an optimal solution to the dual problem, then $\|\boldsymbol{\lambda}^*\|_1 = 1$ must hold; otherwise, the objective is unbounded. Similarly, for every $\boldsymbol{h} \in \Gamma_n$, we must have

$$\sum_{Z \in \Sigma_{\text{out}}} \lambda^*_Z h(Z) - \sum_{\delta \in \Delta} w^*_\delta h(\delta) \le 0$$

in order for the objective to be bounded. This means we can reformulate the Lagrangian dual problem to be exactly the optimization problem in the statement of the lemma. □

Finally, we state and prove the key supporting proposition.

Proposition 6.3 ([5, 6]). *Consider the following optimization problem*

$$\begin{aligned}\min_{\boldsymbol{\lambda}, \boldsymbol{w}} \quad & \prod_{\delta \in \Delta} N_\delta^{w_\delta} \\ \text{s.t.} \quad & \sum_{Z \in \Sigma_{\text{out}}} \lambda_Z \cdot h(Z) \le \sum_{\delta \in \Delta} w_\delta \cdot h(\delta) \text{ is a Shannon-flow inequality} \\ & \|\boldsymbol{\lambda}\|_1 = 1 \text{ and } \boldsymbol{\lambda} \ge \boldsymbol{0}, \boldsymbol{w} \ge \boldsymbol{0}\end{aligned}$$

Then, the following hold:

(i) The problem is solvable in polynomial time in $|2^V|$ and $\sum_{\delta \in \Delta} \log N_\delta$.

(ii) Let $(\boldsymbol{\lambda}, \boldsymbol{w})$ be any optimal solution, and let B denote the optimal *objective value. Then, for any unconditional monotonicity term $\delta \in \Delta$, we have $N_\delta > B$ implies $w_\delta = 0$.*

Proof. Part (i) follows directly from Lemma 6.2. We prove part (ii) next. Fix any optimal solution $(\boldsymbol{\lambda}, \boldsymbol{w})$. Consider the equivalent integral form of the inequality (13) parameterized by $(\mathcal{Z}, \mathcal{D})$. Let $m(Z)$ and $m(\delta)$ denote the multiplicities of $Z \in \Sigma_{\text{out}}$ and $\delta \in \Delta$ in $\mathcal{Z}$ and $\mathcal{D}$ respectively. Then, we have $\lambda_Z = \frac{m(Z)}{|\mathcal{Z}|}$ and $w_\delta = \frac{m(\delta)}{|\mathcal{Z}|}$.

Suppose to the contrary that there is a $\bar{\delta} \in \Delta$ such that $w_{\bar{\delta}} > 0$ and $N_{\bar{\delta}} > B$. Then,

$$B \ge N_{\bar{\delta}}^{w_{\bar{\delta}}} = N_{\bar{\delta}}^{\frac{m(\bar{\delta})}{|\mathcal{Z}|}} > B^{\frac{m(\delta)}{|\mathcal{Z}|}},$$

which implies that $1 \le m(\bar{\delta}) < |\mathcal{Z}|$. From Lemma 2.5, we can construct another integral Shannon-flow inequality parameterized by $(\mathcal{Z}', \mathcal{D}')$, such that $\mathcal{Z}' \subseteq \mathcal{Z}$ and $\mathcal{D}' \subseteq \mathcal{D} - \{\bar{\delta}\}$, with $|\mathcal{Z}'| \ge |\mathcal{Z}| - 1 > 0$. Let $m'(\delta)$ denote the multiplicity of δ in $\mathcal{D}'$ and $m'(Z)$ denote the multiplicity of Z in $\mathcal{Z}'$. Then, $m'(\delta) \le m(\delta)$ for all $\delta \ne \bar{\delta}$, and $m'(\bar{\delta}) \le m(\bar{\delta}) - 1$. Define $\boldsymbol{\lambda}', \boldsymbol{w}'$ as $\lambda'_Z \stackrel{\text{def}}{=} \frac{m'(Z)}{|\mathcal{Z}'|}$ and $w'_\delta \stackrel{\text{def}}{=} \frac{m'(\delta)}{|\mathcal{Z}'|}$, then they form a feasible solution to the optimization problem. The objective value of

$(\boldsymbol{\lambda}', \boldsymbol{w}')$ is

$$\prod_{\delta\in\Delta} N_\delta^{w'_\delta} = \prod_{\delta\in\Delta} N_\delta^{\frac{m'(\delta)}{|\mathcal{Z}'|}} \le \frac{\prod_{\delta\in\Delta} N_\delta^{\frac{m(\delta)}{|\mathcal{Z}'|}}}{N_{\tilde{\delta}}^{\frac{1}{|\mathcal{Z}'|}}} < \frac{B^{\frac{|\mathcal{Z}|}{|\mathcal{Z}'|}}}{B^{\frac{1}{|\mathcal{Z}'|}}} = B^{\frac{|\mathcal{Z}|-1}{|\mathcal{Z}'|}} \le B,$$

contradicting the optimality of $(\boldsymbol{\lambda}, \boldsymbol{w})$. □

6.2 Main Result: Correctness and Runtime of PANDAExpress

Theorem 6.4. *Given a disjunctive datalog rule of the form* (8)*, input database instance over* Σ_{in} *satisfying degree constraints* $(\Delta, \boldsymbol{N})$*, and a Shannon-flow inequality*

$$\sum_{Z\in\Sigma_{\mathsf{out}}} \lambda_Z h(Z) \le \sum_{\delta\in\Delta} w_\delta h(\delta). \tag{38}$$

Then, PANDAExpress can compute a model Σ_{out} *for the DDR* (8) *of size* $\|\Sigma_{\mathsf{out}}\| = O(B)$ *in time* $O(N \log N + B \log N)$ *where* N *is the size of the input database instance, and*

$$B \stackrel{def}{=} \prod_{\delta\in\Delta} N_\delta^{w_\delta}. \tag{39}$$

Proof. Let $(\mathcal{Z}, \mathcal{D})$ be the multisets representing the equivalent integral Shannon-flow inequality (6) of the given inequality (38). For each $\delta = (Y|X) \in \Delta$, let $R \in \Sigma_{\mathsf{in}}$ be a relation where $\max_{\boldsymbol{x}} \deg_R(Y|X = \boldsymbol{x}) \le N_\delta$. Define the following conditional sub-probability measure:

$$p_\delta(\boldsymbol{y}|\boldsymbol{x}) \stackrel{\text{def}}{=} \begin{cases} \frac{1}{N_\delta} & (\boldsymbol{x}, \boldsymbol{y}) \in R_{|X\cup Y} \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

These measures define the set $\mathcal{P}$, which along with $(\mathcal{Z}, \mathcal{D})$ are the initial inputs to the algorithm PANDAExpress. The recursive calls to PANDAExpress form a tree of branching factor at most 2 at each internal node, called the *execution tree*. The leaf nodes are where we return and gather the results, where for every $Z \in \mathcal{Z}$ the output $Q(Z)$ is the union of all $Q(Z)$ constructed at the leaf nodes.

Claim: Throughout the execution, the following invariants hold:

(a) Every pair $(\mathcal{Z}, \mathcal{D})$ passed to PANDAExpress defines a valid Shannon-flow inequality of the form (6).
(b) For every $p_\delta \in \mathcal{P}$ where $\delta = (Y|X)$, and every $\boldsymbol{x} \in \mathrm{Dom}^X$, $p_{Y|X=\boldsymbol{x}}$ is a sub-probability measure.
(c) Every $\mathcal{Z}$ passed to PANDAExpress is not empty.
(d) For every $p_Y \in \mathcal{P}$ (unconditional measure), we have $p_Y(\boldsymbol{y}) > 0 \Rightarrow p_Y(\boldsymbol{y}) \ge 1/B$ for all $\boldsymbol{y} \in \mathrm{Dom}^Y$.
(e) For every tuple $\boldsymbol{t} \in \bowtie\Sigma_{\mathsf{in}}$, there exists a leaf node PANDAExpress$(\mathcal{Z}, \mathcal{D}, \mathcal{P})$ in the execution tree where

$$\prod_{\delta\in\mathcal{D}} p_\delta(\boldsymbol{t}) \ge \frac{1}{B^{|\mathcal{Z}|}}. \tag{41}$$

Assuming the claim holds, we prove that the algorithm is correct and analyze its runtime.

We start with correctness. For any $\boldsymbol{t} \in \bowtie\Sigma_{\mathsf{in}}$, consider the leaf node PANDAExpress$(\mathcal{Z}, \mathcal{D}, \mathcal{P})$ in the execution tree where (41) holds. Since it is a leaf node, there exists $Z \in \mathcal{Z} \cap \mathcal{D}$. From (41) we have $p_Z(\boldsymbol{t}) > 0$, which means $\boldsymbol{t}_Z \in Q(Z)$. Thus, the output is a model of the input DDR.

Next, we analyze the runtime. The length of the path from the root down to the first internal node with 2 children is at most the length of the proof sequence of the Shannon-flow inequality (6). At the branching point, the light branch continues on with the next proof step in the sequence, while the heavy branch resets the inequality by applying the Reset Lemma 2.5. Either way, the potential function $|\mathcal{D}| + |\mathcal{M}| + 3|\mathcal{S}|$ decreases by at least 1, thanks to Theorem 2.4 and Lemma 2.5. Therefore, the depth of the execution tree is at most $|\mathcal{D}| + |\mathcal{M}| + 3|\mathcal{S}|$, which is a query-dependent parameter.

To show the $O((N+B)\log N)$-runtime overall, it is thus sufficient to argue that every call to apply-step takes $O((N+B)\log N)$-time. From invariant (d), every unconditional measure $p_Y \in \mathcal{P}$ has at most B tuples in its support. Therefore, every decomposition and monotonicity step can be implemented in $O((N+B)\log N)$-time because computing the marginal and conditional measures can be done with at most 2 passes over the support of the original measure. The submodularity step is even simpler because it only requires renaming a measure. Finally, the composition step (36) can also be implemented in $O((N+B)\log N)$-time because we can scan over p_X, and for each $\boldsymbol{x}$ in its support, compute the threshold $1/(B \cdot p_X(\boldsymbol{x}))$ to filter the support of $p_{Y|X=\boldsymbol{x}}$. The output measure p_{XY} has at most B tuples in its support because of invariant (d).

It remains to prove the claim. Invariants (a) and (b) hold trivially by design. Invariant (c) holds at the root of the execution tree because $\|\boldsymbol{\lambda}\|_1 = 1$, and it continues to hold afterwards, thanks to the condition $|\mathcal{Z}| > 1$ in line (9) of Algorithm 1.

Invariant (d) also holds by design for every new unconditional measure p_Y created during the execution, from the composition step. Thus, we only need to show that it holds at the start of the algorithm. Without loss of generality, we can assume that the coefficients $(\boldsymbol{w}, \boldsymbol{\delta})$ given in the input Shannon-flow inequality (38) are an optimal solution to the optimization problem in Proposition 6.3, which implies we can assume $\deg_{\Sigma_{\text{in}}}(\delta) \le B$ for every unconditional term $\delta \in \Delta$. Thus, invariant (d) holds at the start.

We prove invariant (e) by induction on the execution tree. We will prove that (e) holds for *every* current execution tree during the execution of the algorithm, not just the last one. At the start, there is only one initial node, where inequality (41) holds as an *equality* for every tuple $\boldsymbol{t} \in \bowtie \Sigma_{\text{in}}$. It is straightforward to verify that the product $\prod_{\delta\in\mathcal{D}} p_\delta(\boldsymbol{t})$ is not reduced by any step other than the composition step in Eq. (36). In the composition step $h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{XY})$, the node $(\mathcal{Z}, \mathcal{D}, \mathcal{P})$ branches into two child nodes: the light child node $(\mathcal{Z}, \mathcal{D}^\ell, \mathcal{P}^\ell)$ and the heavy child node $(\mathcal{Z}^h, \mathcal{D}^h, \mathcal{P}^h)$, where the heavy child node only exists assuming $|\mathcal{Z}| > 1$. Consider a tuple $\boldsymbol{t}$ for which (41) holds at the parent node. We consider two cases:

- If $p_X(\boldsymbol{t}_X) \cdot p_{Y|X}(\boldsymbol{t}_Y|\boldsymbol{t}_X) \ge 1/B$, then $\prod_{\delta\in\mathcal{D}^\ell} p_\delta(\boldsymbol{t}) = \prod_{\delta\in\mathcal{D}} p_\delta(\boldsymbol{t})$, and thus (41) holds at the light child node.
- If $p_X(\boldsymbol{t}_X) \cdot p_{Y|X}(\boldsymbol{t}_Y|\boldsymbol{t}_X) < 1/B$, then we want to show that $\prod_{\delta\in\mathcal{D}^h} p_\delta(\boldsymbol{t}) \ge 1/B^{|\mathcal{Z}^h|}$ for the $(\mathcal{Z}^h, \mathcal{D}^h)$ pair resulted from the reset step, where we know $\mathcal{Z}^h \subseteq \mathcal{Z}$, $|\mathcal{Z}^h| \ge |\mathcal{Z}| - 1$, and $\mathcal{D}^h \subseteq \mathcal{D} - \{(\boldsymbol{X}), (\boldsymbol{Y}|\boldsymbol{X})\}$. We also want to show that $|\mathcal{Z}| > 1$ since, otherwise, the heavy child would not have been created in the first place. Since (41) holds at the parent node, we have

$$1 \ge \prod_{\delta\in\mathcal{D}^h} p_\delta(\boldsymbol{t}) \ge \frac{\prod_{\delta\in\mathcal{D}} p_\delta(\boldsymbol{t})}{p_X(\boldsymbol{t}_X)\cdot p_{Y|X}(\boldsymbol{t}_Y|\boldsymbol{t}_X)} > B \cdot \frac{1}{B^{|\mathcal{Z}|}} \ge \frac{1}{B^{|\mathcal{Z}^h|}}. \tag{42}$$

  This inequality proves invariant (e) for the heavy child node, assuming $|\mathcal{Z}| > 1$. It also proves that $|\mathcal{Z}| > 1$ because if $|\mathcal{Z}| = 1$, then the inequality becomes $1 > 1$, which is a contradiction. (Invariant (c) rules out the case of $|\mathcal{Z}| = 0$.)

□

7 Disjunctive datalog rules, conjunctive queries, and the submodular width

This section provides a brief overview of the notions of fractional hypertree width and submodular width under degree constraints, and how conjunctive queries can be answered in submodular width time via the connection to DDRs. All the materials were implicit or explicit in [5, 6]. However, we adapted the definitions to the bounds presented in this paper which are expressed directly in the input degree constraints, so they are in the primal form instead of the dual form as in [5, 6].

7.1 Tree decompositions and free-connex tree decompositions

A *tree decomposition* of a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a pair (T, χ) where $T = (V(T), E(T))$ is a tree and $\chi : V(T) \to 2^{\mathcal{V}}$ is a mapping that assigns to each node of T a set of vertices of $\mathcal{H}$, called a *bag*, such that: (i) for every hyperedge $S \in \mathcal{E}$, there exists a node $t \in V(T)$ such that $S \subseteq \chi(t)$; and (ii) for every vertex $v \in \mathcal{V}$, the set of nodes $\{t \in V(T) \mid v \in \chi(t)\}$ induces a connected subtree of T.

Consider a conjunctive query (CQ) $Q(F) \coloneq \bigwedge_{S\in\mathcal{E}} R_S(S)$ where $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is its associated hypergraph, $F \subseteq \mathcal{V}$ is the set of free variables of Q, and each R_S is a relation over the set of variables $S \subseteq \mathcal{V}$. Note that we abuse notation and use S to refer to both the hyperedge in $\mathcal{E}$ and the set of variables in the relation R_S. In the notations established in the paper, $\Sigma_{\mathsf{in}} = \{R_S \mid S \in \mathcal{E}\}$.

A *free-connex* tree decomposition [11] of Q is a tree decomposition (T, χ) of its hypergraph $\mathcal{H}$ such that there is a connected subtree of T whose bags contain *all* the free variables F of Q and *no* non-free variables. Note that, if Q was a Boolean CQ or a full CQ, then every tree decomposition of $\mathcal{H}$ is free-connex. Let $\mathsf{TD}(Q)$ denote the set of all free-connex tree decompositions of Q.

7.2 Fractional hypertree width under degree constraints

Suppose Σ_{in} satisfies degree constraints $(\Delta, \boldsymbol{N})$. A free-connex tree decomposition (T, χ) of Q can be thought of as a query plan for answering Q. Each bag $\chi(t)$ defines the following DDR:

$$Q_t(\chi(t)) \coloneq \bigwedge_{S\in\mathcal{E}} R_S(S). \tag{43}$$

The answer to Q can be computed in linear time (and with constant delay) from the answers to all DDRs (43) (which are CQs) [11], one for each bag of (T, χ). The time it takes to answer the DDR (43), from Theorem 6.4 is $O((N + B)\log N)$ where

$$B \stackrel{\text{def}}{=} \min_{\boldsymbol{w}\ge\boldsymbol{0}} \left\{ \prod_{\delta\in\Delta} N_\delta^{w_\delta} \text{ where } h(\chi(t)) \le \sum_{\delta\in\Delta} w_\delta h(\delta) \text{ is a valid Shannon inequality} \right\} \tag{44}$$

To compute this bound, we can solve equivalently the following optimization problem (which is solvable in polynomial time, thanks to Proposition 6.3):

$$\rho^*(t, \Delta, \boldsymbol{N}) \stackrel{\text{def}}{=} \min_{\boldsymbol{w}\ge\boldsymbol{0}} \sum_{\delta\in\Delta} w_\delta \cdot \log_N N_\delta \tag{45}$$

$$\text{subject to } h(\chi(t)) \le \sum_{\delta\in\Delta} w_\delta h(\delta) \text{ is a valid Shannon inequality} \tag{46}$$

The quantity $\rho^*(t, \Delta, \boldsymbol{N})$ is the generalization of the *fractional edge cover number* of a bag t under degree constraints. If all constraints were cardinality constraints, then (46) is exactly the Shearer's inequality.

Hence, if our query plan consists of a single tree decomposition, then the best time we can hope for is parameterized by the generalized fractional hypertree width of Q under degree constraints:

$$\mathsf{fhtw}(Q, \Delta, \boldsymbol{N}) \stackrel{\text{def}}{=} \min_{(T,\chi)\in\mathsf{TD}(Q)} \max_{t\in V(T)} \rho^*(t, \Delta, \boldsymbol{N}). \tag{47}$$

Note that $\mathsf{TD}(Q)$ is the set of all free-connex tree decompositions of Q, and thus this notion generalizes the classical fractional hypertree width to not only handle degree constraints but also non-Boolean and non-full CQs. A direct corollary of Theorem 6.4 is the following:

Corollary 7.1. *Given a CQ Q and degree constraints $(\Delta, \boldsymbol{N})$ on its input relations, we can compute the answer to Q in time $O(N^{fhtw(Q,\Delta,\boldsymbol{N})} \log N + |Q|)$, with constant-delay enumeration of the answers.*

There is a subtle point that our definition of fhtw in (47) above is a smooth generalization of the original fractional hypertree width definition from [24] where the fractional edge covering number is replaced by $\rho^*(t, \Delta, \boldsymbol{N})$. However, the definition of fhtw given in [5, 6] is slightly different because it is based on the dual form of the optimization problem.[4] Nevertheless, thanks to Lemma 6.2, we can see that the definitions are equivalent:

$$\mathsf{fhtw}(Q, \Delta, \boldsymbol{N}) \stackrel{\text{def}}{=} \min_{(T,\chi)\in\mathsf{TD}(Q)} \max_{t\in V(T)} \rho^*(t, \Delta, \boldsymbol{N}) = \min_{(T,\chi)\in\mathsf{TD}(Q)} \max_{t\in V(T)} \max_{\boldsymbol{h}\models(\Delta,\boldsymbol{N})} h(\chi(t))$$

7.3 Submodular width under degree constraints

The fractional hypertree width was derived by minimizing the output size, over all tree decompositions (T, χ) of the following query:

$$\bigwedge_{t\in V(T)} Q_t(\chi(t)) \text{ :- } \bigwedge_{S\in\mathcal{E}} R_S(S). \tag{48}$$

The conjunction in the head means we can evaluate each DDR Q_t independently. Daniel Marx [30] had a brilliant insight that we do not need to use a single tree decomposition, even the best one; instead, we can load-balance the computation over multiple tree decompositions to obtain a better runtime. To do so, we want to answer the following query optimally:

$$\bigvee_{(T,\chi)\in\mathsf{TD}(Q)} \bigwedge_{t\in V(T)} Q_t(\chi(t)) \text{ :- } \bigwedge_{S\in\mathcal{E}} R_S(S). \tag{49}$$

Semantically, query (49) means that, for every satisfying tuple of the query's body, as long as there is one tree decomposition for which all their bag atoms are satisfied, then the tuple is in the output. If we can compute an answer to query (49), then from each tree decomposition, we can enumerate with constant delay their contribution to the answer to Q. This means over all, the answer to Q can also be enumerated with constant delay.

Query (49) is neither a typical CQ nor a typical DDR. The second idea we need is the distributivity of $\vee$ over $\wedge$ to rewrite it from DNF-form to CNF-form. In order to describe this rewriting, we need a numbering of tree decompositions and their bags. We number the tree decompositions in $\mathsf{TD}(Q)$ as (T_i, χ_i) for $i \in [m]$, and let $\boldsymbol{T}$ denote the set of all tuples $\boldsymbol{t} = (t_1, \ldots, t_m)$ where $t_i \in V(T_i)$ is a bag in tree decomposition T_i. Then, we can rewrite query (49) as:

$$\bigwedge_{\boldsymbol{t}\in\boldsymbol{T}} \bigvee_{i\in[m]} Q_{t_i}(\chi_i(t_i)) \text{ :- } \bigwedge_{S\in\mathcal{E}} R_S(S). \tag{50}$$

This query can be answered by answering $|\boldsymbol{T}|$ many DDRs of the form:

$$\bigvee_{i\in[m]} Q_{t_i}(\chi_i(t_i)) \text{ :- } \bigwedge_{S\in\mathcal{E}} R_S(S). \tag{51}$$

[4] As a special case, the fractional edge cover number is the same as the fractional vertex packing number.

From Theorem 6.4, the time to answer this DDR is $O((N+B)\log N)$ where

$$B \stackrel{\text{def}}{=} \min_{\boldsymbol{\lambda}, \boldsymbol{w} \geq \mathbf{0}} \left\{ \prod_{\delta \in \Delta} N_\delta^{w_\delta} \text{ s.t. } \sum_{i \in [m]} \lambda_i \cdot h(\chi_i(t_i)) \leq \sum_{\delta \in \Delta} w_\delta h(\delta) \text{ is a valid Shannon inequality} \right\} \tag{52}$$

To compute this bound, we can solve equivalently the following optimization problem, which is solvable in polynomial time, thanks to Proposition 6.3: (Note that, compared to Eq. (45), we now have an additional variable vector $\boldsymbol{\lambda}$.)

$$\rho^*(\boldsymbol{t}, \Delta, \boldsymbol{N}) \stackrel{\text{def}}{=} \min_{\boldsymbol{\lambda}, \boldsymbol{w} \geq \mathbf{0}} \sum_{\delta \in \Delta} w_\delta \cdot \log_N N_\delta \tag{53}$$

$$\text{subject to } \sum_{i \in [m]} \lambda_i \cdot h(\chi_i(t_i)) \leq \sum_{\delta \in \Delta} w_\delta h(\delta) \text{ is a valid Shannon inequality} \tag{54}$$

Since we have to answer all DDRs indexed by $\boldsymbol{t} \in \boldsymbol{T}$, the time spent is dominated by the worst combination. The submodular width of Q under degree constraints is defined as:

$$\mathsf{subw}(Q, \Delta, \boldsymbol{N}) \stackrel{\text{def}}{=} \max_{\boldsymbol{t} \in \boldsymbol{T}} \rho^*(\boldsymbol{t}, \Delta, \boldsymbol{N}). \tag{55}$$

A direct corollary of Theorem 6.4 is the following:

Corollary 7.2. *Given a CQ Q and degree constraints $(\Delta, \boldsymbol{N})$ on its input relations, we can compute the answer to Q in time $O(N^{subw(Q,\Delta,\boldsymbol{N})} \log N + |Q|)$, with constant-delay enumeration of the answers.*

Similar to the fractional hypertree width case, our definition of the submodular width under degree constraints here is slightly different from that in [5, 6, 30], because we use the primal form of the optimization problem. Thanks to Lemma 6.2, we can see that the definitions are equivalent:

$$\begin{aligned} \mathsf{subw}(Q, \Delta, \boldsymbol{N}) &\stackrel{\text{def}}{=} \max_{\boldsymbol{t} \in \boldsymbol{T}} \rho^*(\boldsymbol{t}, \Delta, \boldsymbol{N}) = \max_{\boldsymbol{t} \in \boldsymbol{T}} \max_{\boldsymbol{h} \models (\Delta, \boldsymbol{N})} \min_{i \in [m]} h(\chi_i(t_i)) \\ &= \max_{\boldsymbol{h} \models (\Delta, \boldsymbol{N})} \max_{\boldsymbol{t} \in \boldsymbol{T}} \min_{i \in [m]} h(\chi_i(t_i)) = \max_{\boldsymbol{h} \models (\Delta, \boldsymbol{N})} \min_{(T, \chi) \in \mathsf{TD}(Q)} \max_{t \in V(T)} h(\chi(t)) \end{aligned}$$

8 Extensions to handle ℓ_p-norm constraints

Abo Khamis et al [29] showed that the PANDA framework can be extended to handle a more general class of constraints, called *ℓ_p-norm constraints*. Let $p \in (0, \infty]$ be a positive integer. Let $\theta = (\boldsymbol{Y}|\boldsymbol{X})_p$ denote what we call a *ℓ_p-norm term* (to generalize monotonicity term). Let R be an input relation over schema Σ. Referring back to notations defined in (10) and (11), we define

$$\deg_R(\theta) := \| \deg_R(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x}) \|_p \qquad \deg_\Sigma(\theta) := \min_{R \in \Sigma} \deg_R(\theta) \tag{56}$$

Given a database instance over schema Σ, we say that Σ satisfies the *ℓ_p-norm constraint* (θ, N_θ), and write $\Sigma \models (\theta, N_\theta)$, if $\deg_\Sigma(\theta) \leq N_\theta$. Similar to $h(\delta)$ and $h(\sigma)$, we define

$$h(\theta) := \frac{1}{p} h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{X}) \qquad \theta = (\boldsymbol{Y}|\boldsymbol{X})_p. \tag{57}$$

Note that if $p = \infty$ then $h(\theta) = h(\delta)$ and $\deg_\Sigma(\theta) = \deg_\Sigma(\delta)$ where $\delta = (\boldsymbol{Y}|\boldsymbol{X})$; in particular, ℓ_p-norm constraints strictly generalize degree constraints. Let Θ denote a set of ℓ_p-norm terms (where p can also vary). Let $\boldsymbol{N} = (N_\theta)_{\theta \in \Theta}$ be a vector of positive real numbers. Given a set Σ_{in} of input relations, we say that $\Sigma_{\mathsf{in}} \models (\Theta, \boldsymbol{N})$ if $\Sigma_{\mathsf{in}} \models (\theta, N_\theta)$ for every $\theta \in \Theta$. The following analog of Theorem 2.7 holds for ℓ_p-norm constraints:

Corollary 8.1 ([29]). *Consider a DDR of the form* (8) *with input schema* Σ_{in}*, and output schema* Σ_{out} *such that* $\Sigma_{\text{in}} \models (\Theta, \boldsymbol{N})$*. Assume that there exist two* non-negative *rational weight vectors* $\boldsymbol{w} \stackrel{def}{=} (w_\theta)_{\theta\in\Theta}$ *and* $\boldsymbol{\lambda} \stackrel{def}{=} (\lambda_Z)_{Z\in\Sigma_{\text{out}}}$ *with* $\|\boldsymbol{\lambda}\|_1 = 1$*, where the following is a Shannon-flow inequality:*

$$\sum_{Z\in\Sigma_{\text{out}}} \lambda_Z \cdot h(Z) \le \sum_{\theta\in\Theta} w_\theta \cdot h(\theta). \tag{58}$$

Then, for any input instance Σ_{in} *of the DDR* (8)*, there exists a model* Σ_{out} *for the DDR for which:*

$$\max_{Z\in\Sigma_{\text{out}}} |Q(Z)| \le \prod_{\theta\in\Theta} N_\theta^{w_\theta} \tag{59}$$

Proof. The proof is virtually identical to that of Corollary 4.5, where we apply Lemma 4.4 but with a different initialization of the probability measures. In particular, inequality (58) *is* a Shannon-flow inequality of the form (13) when we expand it out using the definition of $h(\theta)$ in Eq. (57):

$$\sum_{Z\in\Sigma_{\text{out}}} \lambda_Z \cdot h(Z) \le \sum_{\theta\in\Theta} w_\theta \cdot h(\theta) = \sum_{(Y|X)_p\in\Theta} \frac{w_\theta}{p} \cdot (h(X) + p \cdot h(Y|X)) \,. \tag{60}$$

For each $\theta = (Y|X)_p \in \Theta$, let $R \in \Sigma_{\text{in}}$ be a relation such that $\deg_R(\theta) = \deg_\Sigma(\theta)$. We define two sub-probability measures p_X and $p_{Y|X}$ as follows:

$$p_X(\boldsymbol{x}) := \frac{(\deg_R(Y|X=\boldsymbol{x}))^p}{\|\deg_R(Y|X)\|_p^p} \qquad\qquad p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) := \frac{1}{\deg_R(Y|X=\boldsymbol{x})} \tag{61}$$

Define $B := \prod_{\theta\in\Theta} N_\theta^{w_\theta}$, then from Lemma 4.4 there exist coefficients $\boldsymbol{\lambda} = (\lambda_Z)_{Z\in\Sigma_{\text{out}}}$ with $\|\boldsymbol{\lambda}\|_1 = 1$ and sub-probability measures p_Z, one for each $Z \in \Sigma_{\text{out}}$, such that

$$\prod_{Z\in\Sigma_{\text{out}}} p_Z(\boldsymbol{t})^{\lambda_Z} \ge \prod_{\theta=(Y|X)_p\in\Theta} [p_X(\boldsymbol{t}) \cdot p_{Y|X}(\boldsymbol{t})^p]^{\frac{w_\theta}{p}} \ge \prod_{\theta\in\Theta} \left(\frac{1}{N_\theta^p}\right)^{w_\theta/p} = \frac{1}{B}.$$

The rest of the proof is identical to that of Corollary 4.5. □

The proof above also contains the initialization of the sub-probability measures that allows PANDAExpress to handle ℓ_p-norm constraints, proving the following analog of Theorem 6.4:

Corollary 8.2. *Given a disjunctive datalog rule of the form* (8)*, input database instance over* Σ_{in} *satisfying* ℓ_p*-norm constraints* $(\Theta, \boldsymbol{N})$*, and a Shannon-flow inequality* $\sum_{Z\in\Sigma_{\text{out}}} \lambda_Z h(Z) \le \sum_{\theta\in\Theta} w_\theta h(\theta)$*. Then, PANDAExpress can compute a model* Σ_{out} *for the DDR* (8) *of size* $\|\Sigma_{\text{out}}\| = O(B)$ *in time* $O(N \log N + B \log N)$ *where* N *is the size of the input database instance, and* $B \stackrel{def}{=} \prod_{\theta\in\Theta} N_\theta^{w_\theta}$*.*

9 Conclusion

The PANDAExpress algorithm has two key aspects. The first is the idea of leveraging information theoretic inequalities to derive high-level query plans; and the second is to collect precise skewness information during the execution of the query plan in order to partition the data based on general hyperplane cuts to avoid skews. The partitioning strategies are designed to load-balance the computation among all sub-query plans. Since we were interested in meeting the worst-case bounds defined by the submodular width, we used the threshold parameter B to guide the partitioning of the data.

We believe that this idea of load-balancing data processing amongst different sub-query plans is a powerful and practical idea. An interesting future direction to explore is to see if the load-balancing can be more adaptive to the actual data distribution.

The key remaining open question at the center of the PANDA framework is to bound the length of the proof sequence needed to prove a given Shannon-flow inequality. This problem is closely related to the computational complexity of the polymatroid bound question, which is also open. There are some sub-classes of degree constraints under which a polynomial-length proof sequence exists, and those are *also* the cases where the polymatroid bound is computable in PTIME [26].

Both fhtw and subw are defined as optimization problems over the set $\mathsf{TD}(Q)$ which is of exponential size in query complexity. It is easy to see that, for some queries we do not need to optimize over *all* tree decompositions, but only a small subset of them. It would be interesting to characterize the class of queries for which optimizing over a small subset of tree decompositions suffices to compute fhtw and subw.

Another important question is to study, for which class of input, this framework can be used to answer aggregate queries efficiently. There have been some attempts at defining the counting version of the submodular width [2]; however, that width is higher than necessary for some class of queries, as shown in [13].

Acknowledgments

This work was partially supported by NSF IIS 2314527, NSF SHF 2312195, NSF III 2507117, and a Microsoft professorship.

A Proofs of Some Supporting Results on Proof Sequences

To make this paper more self-contained, we include here the proofs of two supporting results on proof sequences, which were not explicitly stated in [5, 6].

Theorem 2.4 (The Proof Sequence Construction [5, 6]). *Every integral Shannon-flow inequality* (6) *parameterized by* $(\mathcal{Z}, \mathcal{D})$ *with witness* $(\mathcal{M}, \mathcal{S})$ *can be proved by a sequence of steps*

$$\sum_{\delta\in\mathcal{D}} h(\delta) = \sum_{\delta\in\mathcal{D}_0} h(\delta) \geq \sum_{\delta\in\mathcal{D}_1} h(\delta) \geq \cdots \geq \sum_{\delta\in\mathcal{D}_k} h(\delta), \text{ with } \mathcal{D}_k \supseteq \mathcal{Z}$$

where $\sum_{\delta\in\mathcal{D}_i} h(\delta)$ *is turned into* $\sum_{\delta\in\mathcal{D}_{i+1}} h(\delta)$ *by applying exactly one of the following steps:*

- Submodularity: $h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{Y}|\boldsymbol{XZ})$.
- Composition: $h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{XY})$.
- Decomposition: $h(\boldsymbol{XY}) \to h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{X})$.
- Monotonicity: $h(\boldsymbol{XY}) \to h(\boldsymbol{X})$.

Furthermore, $(\mathcal{Z}, \mathcal{D}_i)$ *is an integral Shannon-flow inequality witnessed by* $(\mathcal{M}_i, \mathcal{S}_i)$*, for all* $0 \leq i \leq k$*, and the number of steps is bounded by* $k \leq |\mathcal{D}| + |\mathcal{M}| + 3|\mathcal{S}|$.

Proof of Theorem 2.4. Let $(\mathcal{Z}, \mathcal{D})$ be parameters of the integral Shannon-flow inequality (4) witnessed by $(\mathcal{M}, \mathcal{S})$. We start with $i = 0$: $(\mathcal{D}_0, \mathcal{M}_0, \mathcal{S}_0) := (\mathcal{D}, \mathcal{M}, \mathcal{S})$, and we will iteratively construct $(\mathcal{D}_{i+1}, \mathcal{M}_{i+1}, \mathcal{S}_{i+1})$.

By Proposition 2.2, identity (5) holds. If $\mathcal{Z} \subseteq \mathcal{D}_i$ (in the multiset sense), then we are done; no proof step is needed. So let's assume $\mathcal{Z} \nsubseteq \mathcal{D}_i$.

We first claim that there must be at least one unconditional term $\boldsymbol{W} \in \mathcal{D}_i$ that is *not* in $\mathcal{Z}$. Assume for the sake of contradiction that there is no unconditional term in $\mathcal{D}_i$. Set all (symbolic) variables $h(\boldsymbol{X}) := 1$, $\boldsymbol{X} \neq \emptyset$. Then, $h(\boldsymbol{Y}|\boldsymbol{X})$ is 0 if $\boldsymbol{X} \neq \emptyset$, and 1 otherwise. Moreover, $h(\boldsymbol{Y}|\boldsymbol{X}) \geq 0$ and $h(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X}) \geq 0$. Hence, the LHS of (5) is exactly $|\mathcal{Z}|$ and the RHS is at most the number of unconditional terms in $\mathcal{D}_i$. Therefore, the number of unconditional terms in $\mathcal{D}_i$ must be at least $|\mathcal{Z}|$. If every unconditional term in $\mathcal{D}_i$ is also in $\mathcal{Z}$, then $\mathcal{Z}$ must be the set of unconditional terms in $\mathcal{D}_i$. This contradicts our assumption that $\mathcal{Z} \nsubseteq \mathcal{D}_i$.

Next, take any $\boldsymbol{W} \in \mathcal{D}_i$ that is not in $\mathcal{Z}$. Since identity (5) holds, there must be some term that cancels $h(\boldsymbol{W})$. We consider a few cases:

- If $(\boldsymbol{Y}|\boldsymbol{W}) \in \mathcal{D}_i$ so that $h(\boldsymbol{Y}|\boldsymbol{W})$ cancels $h(\boldsymbol{W})$, then we define
$$\mathcal{D}_{i+1} := \mathcal{D}_i \setminus \{\boldsymbol{W}, (\boldsymbol{Y}|\boldsymbol{W})\} \cup \{\boldsymbol{YW}\}, \qquad \mathcal{M}_{i+1} := \mathcal{M}_i \qquad \mathcal{S}_{i+1} := \mathcal{S}_i$$
and add to the proof sequence the composition step $h(\boldsymbol{W}) + h(\boldsymbol{Y}|\boldsymbol{W}) \to h(\boldsymbol{YW})$.
- If $(\boldsymbol{Y}|\boldsymbol{X}) \in \mathcal{M}_i$ where $\boldsymbol{W} = \boldsymbol{YX}$ so that $-h(\boldsymbol{Y}|\boldsymbol{X})$ cancels $h(\boldsymbol{W})$, then we define
$$\mathcal{D}_{i+1} := \mathcal{D}_i \setminus \{\boldsymbol{W}\} \cup \{\boldsymbol{X}\}, \qquad \mathcal{M}_{i+1} := \mathcal{M}_i \setminus \{(\boldsymbol{Y}|\boldsymbol{X})\}, \qquad \mathcal{S}_{i+1} := \mathcal{S}_i$$
and add to the proof sequence the monotonicity step $h(\boldsymbol{W}) \to h(\boldsymbol{X})$.
- If $(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X}) \in \mathcal{S}_i$ so that $-h(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})$ cancels $h(\boldsymbol{W})$, where $\boldsymbol{W} = \boldsymbol{XY}$, then we define
$$\mathcal{D}_{i+1} := \mathcal{D}_i \setminus \{\boldsymbol{W}\} \cup \{(\boldsymbol{X}), (\boldsymbol{Y}|\boldsymbol{XZ})\}, \qquad \mathcal{M}_{i+1} := \mathcal{M}_i, \qquad \mathcal{S}_{i+1} := \mathcal{S}_i \setminus \{(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})\}$$
and add to the proof sequence two steps: the first is a decomposition step $h(\boldsymbol{W}) \to h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{X})$, and the second is a submodularity step $h(\boldsymbol{Y}|\boldsymbol{X}) \to h(\boldsymbol{Y}|\boldsymbol{XZ})$.

In all cases above, the potential function $|\mathcal{D}| + |\mathcal{M}| + 3|\mathcal{S}|$ decreases by at least 1 per every step added. □

Lemma 2.5 (The Reset Lemma [6]). *Given an integral Shannon-flow inequality* (6) *parameterized by* $(\mathcal{Z}, \mathcal{D})$ *and witnessed by* $(\mathcal{M}, \mathcal{S})$*, let* $\boldsymbol{W} \in \mathcal{D}$ *be an unconditional monotonicity term. Then, we can construct multisets* $\mathcal{Z}', \mathcal{D}', \mathcal{M}', \mathcal{S}'$ *such that the following hold:*

(a) $(\mathcal{Z}', \mathcal{D}')$ *are parameters of another integral Shannon-flow inequality witnessed by* $(\mathcal{M}', \mathcal{S}')$*,*
(b) $\mathcal{Z}' \subseteq \mathcal{Z}$ *and* $\mathcal{D}' \subseteq \mathcal{D} - \{\boldsymbol{W}\}$*, with* $|\mathcal{Z}'| \geq |\mathcal{Z}| - 1$*.*
(c) $|\mathcal{D}'| + |\mathcal{M}'| + 2|\mathcal{S}'| \leq |\mathcal{D}| + |\mathcal{M}| + 2|\mathcal{S}| - 1$*.*

Proof of Lemma 2.5. This proof is very similar to the proof of Theorem 2.4 above. Let $\boldsymbol{W} \in \mathcal{D}$ be any unconditional term. Since (5) holds as an identity, there must be some term that cancels $h(\boldsymbol{W})$. We consider a few cases:

- $\boldsymbol{W} \in \mathcal{Z}$, in this case we remove $\boldsymbol{W}$ from $\mathcal{D}$ and from $\mathcal{Z}$
- $(\boldsymbol{Y}|\boldsymbol{W}) \in \mathcal{D}$ so that $h(\boldsymbol{Y}|\boldsymbol{W})$ cancels $h(\boldsymbol{W})$, then we apply a composition step $h(\boldsymbol{W}) + h(\boldsymbol{Y}|\boldsymbol{W}) \to h(\boldsymbol{YW})$, which means we set
$$\mathcal{D}' := \mathcal{D} \setminus \{\boldsymbol{W}, (\boldsymbol{Y}|\boldsymbol{W})\} \cup \{\boldsymbol{YW}\}$$
and then apply induction to remove $\boldsymbol{YW}$ from $\mathcal{D}'$.
- $(\boldsymbol{Y}|\boldsymbol{X}) \in \mathcal{M}$ where $\boldsymbol{W} = \boldsymbol{XY}$ so that $-h(\boldsymbol{Y}|\boldsymbol{X})$ cancels $h(\boldsymbol{W})$, then we apply a monotonicity step $h(\boldsymbol{W}) \to h(\boldsymbol{X})$, which means we set
$$\mathcal{D}' := \mathcal{D} \setminus \{\boldsymbol{W}\} \cup \{\boldsymbol{X}\}, \qquad \mathcal{M}' := \mathcal{M} \setminus \{(\boldsymbol{Y}|\boldsymbol{X})\}$$
while keeping $\mathcal{S}$ unchanged. Then, we apply induction to remove $\boldsymbol{X}$ from $\mathcal{D}'$.
- $(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X}) \in \mathcal{S}$ so that $-h(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})$ cancels $h(\boldsymbol{W})$, where $\boldsymbol{W} = \boldsymbol{XY}$; then note that
$$h(\boldsymbol{W}) - h(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X}) = h(\boldsymbol{XY}) - \big(h(\boldsymbol{XY}) + h(\boldsymbol{XZ}) - h(\boldsymbol{X}) - h(\boldsymbol{XYZ})\big) = h(\boldsymbol{XYZ}) - h(\boldsymbol{Z}|\boldsymbol{X}).$$
In this case we set
$$\mathcal{D}' := \mathcal{D} \setminus \{\boldsymbol{W}\} \cup \{(\boldsymbol{XYZ})\}, \qquad \mathcal{M}' := \mathcal{M} \cup \{(\boldsymbol{Z}|\boldsymbol{X})\}, \qquad \mathcal{S}' := \mathcal{S} \setminus \{(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})\}$$
and then apply induction to remove $\boldsymbol{XYZ}$ from $\mathcal{D}'$.

Note that in each of the cases above, the potential function $|\mathcal{D}| + |\mathcal{M}| + 2|\mathcal{S}|$ decreases by at least 1 per every step. □

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases.* Addison-Wesley. http://webdam.inria.fr/Alice/

[2] Mahmoud Abo Khamis, Ryan R. Curtin, Benjamin Moseley, Hung Q. Ngo, Xuanlong Nguyen, Dan Olteanu, and Maximilian Schleich. 2020. Functional Aggregate Queries with Additive Inequalities. *ACM Trans. Database Syst.* 45, 4, Article 17 (dec 2020), 41 pages. doi:10.1145/3426865

[3] Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. 2016. FAQ: Questions Asked Frequently. In *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*, Tova Milo and Wang-Chiew Tan (Eds.). ACM, 13–28. doi:10.1145/2902251.2902280

[4] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2016. Computing Join Queries with Functional Dependencies. In *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*, Tova Milo and Wang-Chiew Tan (Eds.). ACM, 327–342. doi:10.1145/2902251.2902289

[5] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2017. What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another?. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts (Eds.). ACM, 429–444. doi:10.1145/3034786.3056105 Extended version available at http://arxiv.org/abs/1612.02503.

[6] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2025. PANDA: Query Evaluation in Submodular Width. *TheoretiCS* Volume 4, Article 12 (Apr 2025). doi:10.46298/theoretics.25.12

[7] Noga Alon. 1981. On the number of subgraphs of prescribed type of graphs with a given number of edges. *Israel J. Math.* 38, 1-2 (1981), 116–130. doi:10.1007/BF02761855

[8] Noga Alon, Raphael Yuster, and Uri Zwick. 1997. Finding and Counting Given Length Cycles. *Algorithmica* 17, 3 (1997), 209–223. doi:10.1007/BF02523189

[9] Albert Atserias, Martin Grohe, and Dániel Marx. 2008. 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.* IEEE Computer Society, 739–748. doi:10.1109/FOCS.2008.43

[10] Albert Atserias, Martin Grohe, and Dániel Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SIAM J. Comput.* 42, 4 (2013), 1737–1767. doi:10.1137/110859440

[11] Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. 2007. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4646)*, Jacques Duparc and Thomas A. Henzinger (Eds.). Springer, 208–222. doi:10.1007/978-3-540-74915-8_18

[12] Karl Bringmann and Egor Gorbachev. 2024. A Fine-grained Classification of Subquadratic Patterns for Subgraph Listing and Friends. *CoRR* abs/2404.04369 (2024). arXiv:2404.04369 doi:10.48550/ARXIV.2404.04369

[13] Karl Bringmann and Egor Gorbachev. 2025. A Fine-Grained Classification of Subquadratic Patterns for Subgraph Listing and Friends. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, Michal Koucký and Nikhil Bansal (Eds.). ACM, 2145–2156. doi:10.1145/3717823.3718141

[14] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison (Eds.). ACM, 77–90. doi:10.1145/800105.803397

[15] F. R. K. Chung, R. L. Graham, P. Frankl, and J. B. Shearer. 1986. Some intersection theorems for ordered sets and graphs. *J. Combin. Theory Ser. A* 43, 1 (1986), 23–37. doi:10.1016/0097-3165(86)90019-1

[16] Mina Dalirrooyfard, Surya Mathialagan, Virginia Vassilevska Williams, and Yinzhan Xu. 2024. Towards Optimal Output-Sensitive Clique Listing or: Listing Cliques from Smaller Cliques. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, Bojan Mohar, Igor Shinkar, and Ryan O'Donnell (Eds.). ACM, 923–934. doi:10.1145/3618260.3649663

[17] Mina Dalirrooyfard, Thuy-Duong Vuong, and Virginia Vassilevska Williams. 2021. Graph Pattern Detection: Hardness for all Induced Patterns and Faster Noninduced Cycles. *SIAM J. Comput.* 50, 5 (2021), 1627–1662. doi:10.1137/20M1335054

[18] Bailu Ding, Vivek R. Narasayya, and Surajit Chaudhuri. 2024. Extensible Query Optimizers in Practice. *Found. Trends Databases* 14, 3-4 (2024), 186–402. doi:10.1561/1900000077

[19] Friedrich Eisenbrand and Fabrizio Grandoni. 2004. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.* 326, 1-3 (2004), 57–67. doi:10.1016/J.TCS.2004.05.009

[20] Ehud Friedgut. 2004. Hypergraphs, entropy, and inequalities. *Amer. Math. Monthly* 111, 9 (2004), 749–760. doi:10.2307/4145187

[21] Ehud Friedgut and Jeff Kahn. 1998. On the number of copies of one hypergraph in another. *Israel J. Math.* 105 (1998), 251–256. doi:10.1007/BF02780332

[22] Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. 2016. Hypertree Decompositions: Questions and Answers. In *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*, Tova Milo and Wang-Chiew Tan (Eds.). ACM, 57–74. doi:10.1145/2902251.2902309

[23] Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. 2012. Size and Treewidth Bounds for Conjunctive Queries. *J. ACM* 59, 3 (2012), 16:1–16:35. doi:10.1145/2220357.2220363

[24] Martin Grohe and Dániel Marx. 2006. 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*. ACM Press, 289–298. http://dl.acm.org/citation.cfm?id=1109557.1109590

[25] Martin Grohe and Dániel Marx. 2014. Constraint Solving via Fractional Edge Covers. *ACM Trans. Algorithms* 11, 1 (2014), 4:1–4:20. doi:10.1145/2636918

[26] Sungjin Im, Benjamin Moseley, Hung Q. Ngo, and Kirk Pruhs. 2025. Efficient Algorithms for Cardinality Estimation and Conjunctive Query Evaluation With Simple Degree Constraints. *Proc. ACM Manag. Data* 3, 2 (2025), 96:1–96:26. doi:10.1145/3725233

[27] Alon Itai and Michael Rodeh. 1977. Finding a Minimum Circuit in a Graph. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison (Eds.). ACM, 1–10. doi:10.1145/800105.803390

[28] Mahmoud Abo Khamis, Xiao Hu, and Dan Suciu. 2025. Fast Matrix Multiplication meets the Submodular Width. *Proc. ACM Manag. Data* 3, 2 (2025), 98:1–98:26. doi:10.1145/3725235

[29] Mahmoud Abo Khamis, Vasileios Nakos, Dan Olteanu, and Dan Suciu. 2024. Join Size Bounds using $\mathrm{l_p}$-Norms on Degree Sequences. *Proc. ACM Manag. Data* 2, 2 (2024), 96. doi:10.1145/3651597

[30] Dániel Marx. 2013. Tractable Hypergraph Properties for Constraint Satisfaction and Conjunctive Queries. *J. ACM* 60, 6 (2013), 42:1–42:51. doi:10.1145/2535926

[31] Hung Q. Ngo. 2018. Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Houston, TX, USA) *(PODS '18)*. Association for Computing Machinery, New York, NY, USA, 111–124. doi:10.1145/3196959.3196990

[32] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2012. Worst-case optimal join algorithms: [extended abstract]. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini (Eds.). ACM, 37–48. doi:10.1145/2213556.2213565

[33] Hung Q. Ngo, Christopher Ré, and Atri Rudra. 2013. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Rec.* 42, 4 (2013), 5–16. doi:10.1145/2590989.2590991

[34] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1*, Philip A. Bernstein (Ed.). ACM, 23–34. doi:10.1145/582095.582099

[35] Dan Suciu. 2023. Applications of Information Inequalities to Database Theory Problems. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*. IEEE, 1–30. doi:10.1109/LICS56636.2023.10175769

[36] T. Tao. 2021. *An Introduction to Measure Theory*. American Mathematical Society. https://books.google.com/books?id=k0lDEAAAQBAJ

[37] Todd L. Veldhuizen. 2014. Triejoin: A Simple, Worst-Case Optimal Join Algorithm. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, Nicole Schweikardt, Vassilis Christophides, and Vincent Leroy (Eds.). OpenProceedings.org, 96–106. doi:10.5441/002/icdt.2014.13

[38] Yilei Wang. 2022. Personal Communication.

[39] Haozhe Zhang, Christoph Mayer, Mahmoud Abo Khamis, Dan Olteanu, and Dan Suciu. 2025. LpBound: Pessimistic Cardinality Estimation Using ℓ_p-Norms of Degree Sequences. *Proc. ACM Manag. Data* 3, 3 (2025), 184:1–184:27. doi:10.1145/3725321

[40] Yihong Zhang, Yisu Remy Wang, Max Willsey, and Zachary Tatlock. 2022. Relational e-matching. *Proc. ACM Program. Lang.* 6, POPL (2022), 1–22. doi:10.1145/3498696

Received December 2025; accepted February 2026