TheoretiCS

PANDA: Query Evaluation in Submodular Width

Received Jun 5, 2024
Revised Jan 7, 2025
Accepted Feb 28, 2025
Published Apr 30, 2025

Key words and phrases
Shannon inequalities, submodular width, disjunctive datalog, query evaluation, degree constraints

Mahmoud Abo Khamis [a]
Hung Q. Ngo [a]
Dan Suciu [b]

a Relational AI, Berkeley, USA

b University of Washington, Seattle, USA

ABSTRACT. In recent years, several information-theoretic upper bounds have been introduced on the output size and evaluation cost of database join queries. These bounds vary in their power depending on both the type of statistics on input relations and the query plans that they support. This motivated the search for algorithms that can compute the output of a join query in times that are bounded by the corresponding information-theoretic bounds. In this paper, we describe PANDA, an algorithm that takes a Shannon-inequality that underlies the bound, and translates each proof step into an algorithmic step corresponding to some database operation. PANDA computes answers to a conjunctive query in time given by the submodular width plus the output size of the query. The version in this paper represents a significant simplification of the original version in [6].

1. Introduction

Answering *conjunctive queries* efficiently is a fundamental problem in the theory and practice of database management, graph algorithms, logic, constraint satisfaction, and graphical model inference, among others [4, 19, 22, 1]. In a *full conjunctive query*, the input is a set of relations (or tables, or constraints), each with a set of attributes (or variables), and the task is to list all *satisfying assignments*, i.e., assignments to all variables that simultaneously satisfy all input relations. Each such assignment is called an *output tuple*, and their number is the *output size*.

A preliminary version of this article appeared at PODS 2017 [6].
Part of this work was conducted while the authors participated in the Fall 2023 *Simons Program on Logic and Algorithms in Databases and AI*. We thank the anonymous reviewers for many insightful comments.

Cite as Mahmoud Abo Khamis, Hung Q. Ngo, Dan Suciu. PANDA: Query Evaluation in Submodular Width. TheoretiCS, Volume 4 (2025), Article 12, 1-42.
https://theoretics.episciences.org
DOI 10.46298/theoretics.25.12

For example, the query

$$Q(a,b,c) \text{ :- } E(a,b) \wedge E(b,c) \wedge E(c,a) \tag{1}$$

is a full conjunctive query asking for the list of all triangles in a (directed) graph with edge relation E. In contrast, in a *Boolean conjunctive query*, we only ask whether one such assignment exists. The query

$$Q() \text{ :- } E(a,b) \wedge E(b,c) \wedge E(c,a)$$

asks whether there is a triangle in the graph. More generally, in a proper *conjunctive query*, any subset of variables can occur in the head of the query. These are called *free variables*. For example, the query

$$Q(a) \text{ :- } E(a,b) \wedge E(b,c) \wedge E(c,a)$$

asks for the list of all nodes a that are part of a triangle. Other variants of the query evaluation problem include counting output tuples, performing some aggregation over them, or enumerating them under some restrictions.

In the case of a full conjunctive query, the runtime of any algorithm is at least as large as the size of the output. This has motivated the study of upper bounds on the sizes of the query outputs. The corresponding graph-theoretic problem is to bound the number of homomorphic images of a small graph within a larger graph; this problem has a long history in graph theory and isoperimetric inequalities [27, 7, 16, 13, 31, 18]. One such bound is the *AGM bound*, which is a formula that, given only the cardinalities of the input relations, returns an upper bound on the output size of the query [10]. Moreover, the bound is tight, meaning that there exist relations of the given cardinalities where the query's output has size equal to the AGM bound (up to a query-dependent factor). This immediately implies that no algorithm can run in time lower than the AGM bound in the worst-case. Thus, an algorithm that runs in this time is called a *Worst-Case Optimal Join* algorithm. The AGM bound for query (1) is $O(|E|^{3/2})$; algorithms for listing all triangles within this amount of time has been known for decades [24]. For general full conjunctive queries, Grohe and Marx [23], and Atserias, Grohe, and Marx [10, 9] devised a join-project query plan that can compute the output of a full conjunctive query to within a linear factor of the AGM bound, which is very close to optimality. They also showed that a join-only query plan cannot achieve this bound. A few years later, a new class of join algorithms, not based on join and project operators, was able to achieve the AGM bound [36, 32] and thus achieve worst-case optimality.

In practice, especially in database systems, the input cardinalities are not sufficient to model what we know about the data, and definitely not sufficient to predict how good or bad a query plan is. Other data characteristics such as functional dependencies and distinct value counts are often collected and used to optimize queries [30, 25]; furthermore, practical queries often have "relations" that are infinite in pure cardinalities. For example, the output size of this

query

$$Q(a,b) \text{ :- } R(a) \wedge S(b) \wedge a + b = 10$$

is obviously at most $\min\{|R|, |S|\}$, but the AGM bound is $|R| \cdot |S|$. The relation $a + b = 10$ has infinite cardinality. There has thus been a line of research to strengthen the AGM bound to account for increasingly finer classes of input statistics. Specifically, Gottlob, Lee, Valiant and Valiant [21, 20] applied the entropy argument to derive a bound on the output size of a full conjunctive query under general functional dependencies. Their bound, generalizing the AGM-bound, is an optimization problem whose constraints are the Shannon inequalities. This idea was a seed for a series of works that extended the entropy argument to account for finer classes of constraints, including degree constraints [5, 6, 3], and ℓ_p-norm bounds on degree sequences [3].

Designing WCOJ algorithms that match these stronger bounds becomes harder since the bounds under more constraints are tighter. An algorithm called CSMA is described in [5], which only accounts for relation cardinalities and functional dependencies.

In database management systems jargon, a WCOJ algorithm is a *multi-way join operator*; this operator, for some classes of queries, is asymptotically faster than the traditional binary join operator [32]. Given a complicated conjunctive query, however, before applying a WCOJ algorithm, it is often beneficial to come up with a *query plan* that decomposes the query into simpler subqueries. Query plans guided by (hyper)tree decompositions have proved to be very useful both in theory and in practice [19]. In particular, query plans represented by a single tree decomposition can be used to answer a Boolean conjunctive query or a full conjunctive query in time bounded by $O(N^{\mathsf{fhtw}} + |\text{output}|)$, where fhtw is the *fractional hypertree width* [22] of the query, and $|\text{output}|$ is the size of the output. This runtime is sensitive to the output size, and thus it is more adaptive than a single WCOJ application.[1] For example, using tree-decomposition-based query plans, we can identify whether a large graph with N edges contains the homomorphic images of a k-cycle in time $O(N^2)$ (assuming a constant k).

Motivated by finding the fixed-parameter tractability boundary for Boolean conjunctive queries, or equivalently *constraint satisfaction problems*, under the regime of unbounded-arity inputs, Marx [28] came up with a beautifully novel idea: instead of fixing a single query plan (i.e., tree-decomposition) up front, we can consider multiple query plans, and partition the data to make use of different plans for different parts of the data. The improved complexity measure that Marx introduced is called the *submodular width*, subw, which is always less than or equal to fhtw.

The subw algorithm from Marx [28] has some limitations. First, it assumes all input relations have the same cardinality N; in particular, it is not known how to define the width and design similar algorithms under functional dependencies or degree constraints. Second,

1 It should be noted, however, that the sub-queries of the plan are still being answered with a WCOJ operator.

the runtime of the algorithm is *not* $O(N^{\mathsf{subw}})$, but a polynomial in this quantity. Third, the subw-notion and the algorithm were not defined for general conjunctive queries with arbitrary free variables.

Contributions. This paper connects all three of these lines of research: WCOJ algorithms, fractional hypertree width, and submodular width into a single framework, while dealing with arbitrary input degree constraints (which is a superset of functional dependencies and cardinality constraints), *and* arbitrary conjunctive queries. The bridge that connects these algorithms and concepts is information theory. In particular, our contributions include the following:

We show how a generalized version of the classic *Shearer's inequality* [18] can be used to derive upper bounds on the output size of a *disjunctive datalog rule* (DDR), which is a generalization of a conjunctive query. The upper bound is a generalization of the AGM bound to include degree constraints and DDRs. The introduction of DDR and its information theoretic output size bound in studying conjunctive queries is our first major technical novelty. DDRs are interesting in their own right. They form the building blocks of *disjunctive datalog* [17], which is a significant extension of datalog. Disjunctive datalog has a long history: it emerged in logic programming [26, 29], and is used for knowledge representation, representing incomplete information, and constraint satisfaction. Our bound on the output size of a DDR represents a bound on the output size of the *minimal model of the disjunctive datalog program* consisting of that single rule.

Next, we show that certain symbolic manipulations of the information inequality can be converted into a query evaluation plan for the DDR that runs in time bounded by the predicted upper bound. This idea of converting a proof of an information inequality into a query evaluation plan is our second major technical novelty. The algorithm is called PANDA, which stands for "**P**roof-**A**ssisted e**N**tropic **D**egree-**A**ware". In particular, PANDA is worst-case optimal for DDRs under arbitrary degree constraints. Even when restricted to only conjunctive queries, this is already beyond the state of the art in WCOJ algorithms because previous WCOJ algorithms cannot meet the tightened bounds under degree constraints.

Lastly, we explain how to define the notions of fhtw and subw under degree constraints and for conjunctive queries with arbitrary free variables. We show how PANDA can be used to answer arbitrary conjunctive queries with arbitrary degree constraints, in time bounded by $\tilde{O}(N^{\mathsf{subw}} + |\text{output}|)$, where $\tilde{O}$ hides a polylogarithmic factor in N. These results close the gaps left by Marx' work. For example, with PANDA, the k-cycle query can now be answered in $\tilde{O}(N^{2-1/\lceil k/2 \rceil})$ time, which is sub-quadratic, and matches the specialized cycle detection algorithm from Alon, Yuster, and Zwick [8].

The results in this paper were first announced in a conference paper [6]. The current paper makes several significant improvements:

— In [6], we used both the primal and the dual linear program to guide PANDA: the primal gives an optimal polymatroid $\boldsymbol{h}$, while the dual represents the basic Shannon inequalities. In the new version, we use only the dual, which significantly simplifies the algorithm. The algorithm is described only in terms of an information inequality and its proof (called a *witness*), which correspond precisely to a feasible solution to the dual program. We only need to describe the primal and dual programs later, in Sec. 6, where we introduce the degree-aware submodular width, which is defined in terms of the primal.
— Previously, we needed a *proof sequence* to drive the algorithm; it was difficult to prove that a proof sequence exists; for example, no proof sequence existed in our earlier framework [5]. In the new version, we describe PANDA without the need for an explicit proof sequence, which again simplifies it. If needed, a proof sequence can still be extracted from the new version of the algorithm.
— One difficulty in the earlier presentation of PANDA was the need to recompute the proof sequence after a reset step. This is no longer necessary here.

Paper Outline This paper is organized as follows. Section 2 presents background concepts needed to understand the techniques and results of the paper; in particular, it introduces *disjunctive datalog rules* (DDR), a generalization of conjunctive queries, reviews necessary background on *information theory*, and defines the class of statistics on input relations that the PANDA algorithm supports, which are called *degree constraints*. Section 3 discusses the class of information inequalities that are at the center of our work, where they are used to both derive upper bounds on the output size of a query and guide the PANDA algorithm. Section 4 states the main algorithmic result, which says that the PANDA algorithm meets this information-theoretic upper bound *if* the bound is a Shannon inequality, i.e., it does not involve non-Shannon inequalities [40, 39]. Section 5 presents the core PANDA algorithm, which constructs a step-by-step proof of the Shannon inequality, and converts each step into a database operation. Section 6 defines the degree-constraint aware submodular width, and shows how to use disjunctive datalog rules to compute a conjunctive query in time given by the submodular width. We conclude in Section 7.

2. Preliminaries

2.1 Database instances and conjunctive queries (CQ)

Fix a set $\boldsymbol{V}$ of variables (or attributes). An *atom* is an expression of the form $R(\boldsymbol{X})$ where R is a relation name and $\boldsymbol{X} \subseteq \boldsymbol{V}$ is a set of attributes. A *schema*, Σ, is a set of atoms. We shall assume throughout that distinct atoms in Σ have distinct attribute sets. If R is a relation name in Σ, we write $\mathsf{vars}(R)$ for its attribute set, and define $\mathsf{vars}(\Sigma) \stackrel{\text{def}}{=} \boldsymbol{V}$ to be the set of all attributes in the schema Σ.

Given a countably infinite domain Dom, we use $\mathsf{Dom}^{\boldsymbol{X}}$ to denote the set of tuples with attributes $\boldsymbol{X} \subseteq \boldsymbol{V}$. A Σ-*instance* is a map D that assigns to each relation name R in Σ a finite subset $R^D \subseteq \mathsf{Dom}^{\mathsf{vars}(R)}$. Technically, we should use $\Sigma^D \stackrel{\text{def}}{=} (R^D)_{R\in\Sigma}$ to denote the Σ-instance; however, to reduce notational burden, instead of writing R^D and Σ^D, we will often write R and Σ when the instance is clear from the context. Given $\boldsymbol{X} \subseteq \boldsymbol{V}$ and a tuple $\boldsymbol{t} \in \mathsf{Dom}^{\boldsymbol{V}}$, we write $\pi_{\boldsymbol{X}}(\boldsymbol{t})$ to denote the *projection* of $\boldsymbol{t}$ onto the variables $\boldsymbol{X}$.

The *full natural join* (or *full join* for short) of the Σ-instance is the set of tuples $\boldsymbol{t} \in \mathsf{Dom}^{\boldsymbol{V}}$ that satisfy all atoms in Σ:

$$\bowtie\Sigma \stackrel{\text{def}}{=} \{\boldsymbol{t} \in \mathsf{Dom}^{\boldsymbol{V}} \mid \pi_{\mathsf{vars}(R)}(\boldsymbol{t}) \in R, \forall R \in \Sigma\} \tag{2}$$

This set of tuples is sometimes called in the literature the *universal table* of the instance Σ.

Given a schema Σ, a *conjunctive query* is the expression

$$Q(\boldsymbol{F}) \text{ :- } \bigwedge_{R(\boldsymbol{X})\in\Sigma} R(\boldsymbol{X}) \tag{3}$$

where $\boldsymbol{F} \subseteq \boldsymbol{V}$ is called the set of *free variables*, and $Q(\boldsymbol{F})$ is the *head atom* of the query. Atoms in Σ are called the *body atoms* of the query. The output $Q(\boldsymbol{F})$ of an input instance Σ is the projection of the full join (2) onto the free variables $\boldsymbol{F}$: $Q(\boldsymbol{F}) \stackrel{\text{def}}{=} \pi_{\boldsymbol{F}}(\bowtie\Sigma)$.

When $\boldsymbol{F} = \boldsymbol{V}$, we call the query a *full conjunctive query*. When $\boldsymbol{F} = \emptyset$, we call the query a *Boolean conjunctive query*, whose answer $Q()$ is either true or false, and it is true if and only if the full join (2) is non-empty.

Our complexity results are expressed in terms of *data complexity*; in particular, we consider the number of atoms and variables to be a constant, i.e., $|\Sigma| + |\boldsymbol{V}| = O(1)$, and the complexity is a function of the instance size. We define the *size* of a Σ-instance as:

$$\|\Sigma\| \stackrel{\text{def}}{=} \sum_{R(\boldsymbol{X})\in\Sigma} |R|. \tag{4}$$

The notation $\|\Sigma\|$ is used in part to distinguish it from $|\Sigma|$ which counts the number of atoms in Σ.

2.2 Tree decompositions and free-connex queries

Consider a conjunctive query in the form (3). A *tree decomposition* of Q is a pair (T, χ), where T is a tree and $\chi : \mathsf{nodes}(T) \to 2^{\mathsf{vars}(Q)}$ is a map from the nodes of T to subsets of $\mathsf{vars}(Q)$ that satisfies the following properties: for all atoms $R(\boldsymbol{X})$ in Σ, there is a node $t \in \mathsf{nodes}(T)$ such that $\boldsymbol{X} \subseteq \chi(t)$; and, for any variable $X \in \mathsf{vars}(Q)$, the set $\{t \mid X \in \chi(t)\}$ forms a connected subtree of T. Each set $\chi(t)$ is called a *bag* of the tree-decomposition, and we will assume w.l.o.g. that the bags are distinct, i.e., $\chi(t) \neq \chi(t')$ when $t \neq t'$ are nodes in T.

A *free-connex* tree decomposition for Q is a tree-decomposition for Q with an additional property that there is a connected subtree T' of T for which $\boldsymbol{F} = \bigcup_{t\in\mathsf{nodes}(T')} \chi(t)$. The query Q

is *free-connex acyclic* iff there is a free-connex tree decomposition (T, χ) in which every bag is covered by an input atom; namely, for every $t \in \mathsf{nodes}(T)$, there exists an input atom $R(\boldsymbol{X}) \in \Sigma$ where $\chi(t) \subseteq \boldsymbol{X}$.

The following result is well-known [37, 11].

LEMMA 2.1. *If Q is a free-connex acyclic conjunctive query of the form* (3)*, then we can compute its output in time $\tilde{O}(\|\Sigma\| + |Q(\boldsymbol{F})|)$. In particular, after a preprocessing time of $\tilde{O}(\|\Sigma\|)$, we can list the output tuples one by one with constant-delay between them.*

In particular, for this class of queries, the runtime is the best one can hope for: input size plus output size.

2.3 Disjunctive Datalog rules (DDR)

Disjunctive Datalog rules [17] are a generalization of conjunctive queries, where the head of the query can be a disjunction, formalized as follows. Let Σ_{in} and Σ_{out} be two schemas, called input and output schema respectively. We associate to these two schemas the following *disjunctive Datalog rule* (DDR)

$$\bigvee_{Q(\boldsymbol{Z}) \in \Sigma_{\mathsf{out}}} Q(\boldsymbol{Z}) \text{ :- } \bigwedge_{R(\boldsymbol{X}) \in \Sigma_{\mathsf{in}}} R(\boldsymbol{X}) \tag{5}$$

The DDR is uniquely defined by the two schemas Σ_{in} and Σ_{out}. The syntax in (5) does not add any new information, but is intended to be suggestive for the following semantics:

DEFINITION 2.2. Let Σ_{in} be an input instance. A *model* (or *feasible output*) for the rule (5) is an instance Σ_{out}, such that the following condition holds: for every tuple $\boldsymbol{t} \in \bowtie \Sigma_{\mathsf{in}}$, there is an output atom $Q(\boldsymbol{Z}) \in \Sigma_{\mathsf{out}}$ for which $\pi_{\boldsymbol{Z}}(\boldsymbol{t}) \in Q$. Similar to (4), the *size* of the output instance Σ_{out} is defined as

$$\|\Sigma_{\mathsf{out}}\| \stackrel{\text{def}}{=} \sum_{Q(\boldsymbol{Z}) \in \Sigma_{\mathsf{out}}} |Q|. \tag{6}$$

A model is *minimal* if we cannot remove a tuple from any output relation without violating the feasibility condition.

In English, a feasible output needs to store each tuple $\boldsymbol{t} \in \bowtie \Sigma_{\mathsf{in}}$ in at least one of the output relations $Q(\boldsymbol{Z}) \in \Sigma_{\mathsf{out}}$. The query evaluation problem is to compute a minimal model. Note that a conjunctive query is a disjunctive Datalog rule where Σ_{out} has a single atom.

DDRs are interesting. We illustrate the concept here with a few examples.

EXAMPLE 2.3. Consider the following DDR,

$$Q(X, Z) \text{ :- } R(X, Y) \wedge S(Y, Z)$$

where $\Sigma_{\text{in}} = \{R(X,Y), S(Y,Z)\}$ and $\Sigma_{\text{out}} = \{Q(X,Z)\}$. A model (or feasible output) to the DDR is any superset of $\pi_{XZ}(R \bowtie S)$. Consider now the following DDR:

$$A(X) \vee B(Y) \text{ :- } R(X,Y)$$

One model is $A := \pi_X(R)$, $B := \emptyset$. Another model is $A := \emptyset$, $B := \pi_Y(R)$. Both are minimal. Many other models exist.

A non-trivial DDR is the following:

$$A(X,Y,Z) \vee B(Y,Z,W) \text{ :- } R(X,Y) \wedge S(Y,Z) \wedge U(Z,W) \tag{7}$$

To compute the rule, for each tuple (x,y,z,w) in the full join, we must either insert (x,y,z) into A, or insert (y,z,w) into B. The output size is the larger of the resulting relations A and B. We shall see later in the paper that, for this rule, there is a model of size $O(\sqrt{|R| \cdot |S| \cdot |U|})$, which is a non-trivial result. ◆

2.4 Entropic vectors and polymatroids

For general background on information theory and polymatroids, we refer the reader to [38]. Given a discrete (tuple of) random variable(s) $\boldsymbol{X}$ over a domain $\mathsf{Dom}(\boldsymbol{X})$ with distribution p, the (Shannon) *entropy* of $\boldsymbol{X}$ is defined as

$$h(\boldsymbol{X}) \stackrel{\text{def}}{=} -\sum_{\boldsymbol{x} \in \mathsf{Dom}(\boldsymbol{X})} p(\boldsymbol{X} = \boldsymbol{x}) \log p(\boldsymbol{X} = \boldsymbol{x})$$

The *support* of the distribution is the set of all $\boldsymbol{x}$ where $p(\boldsymbol{X} = \boldsymbol{x}) > 0$. We will only work with distributions of finite support. Let N be the support size, then it is known that $0 \le h(\boldsymbol{X}) \le \log N$. The upper bound follows from the concavity of h and Jensen's inequality. Moreover, $h(\boldsymbol{X}) = 0$ iff X is *deterministic*, i.e., it has a singleton support, and $h(\boldsymbol{X}) = \log N$ iff X is *uniform*, meaning that $p(\boldsymbol{X} = \boldsymbol{x}) = 1/N$ for all $\boldsymbol{x}$ in the support.

If $\boldsymbol{V}$ is a set of jointly distributed random variables, then the vector $\boldsymbol{h} = (h(\boldsymbol{X}))_{\boldsymbol{X} \subseteq \boldsymbol{V}} \in \mathbb{R}_+^{2^{\boldsymbol{V}}}$ is called an *entropic vector*.[2] Implicitly, $h(\emptyset) = 0$; thus, this vector is actually of dimension $2^{|\boldsymbol{V}|} - 1$. We will often write $\boldsymbol{XY}$ for the union $\boldsymbol{X} \cup \boldsymbol{Y}$. In particular, $h(\boldsymbol{XY}) = h(\boldsymbol{X} \cup \boldsymbol{Y})$.

Starting with Shannon himself, the study of linear functions on entropic vectors has been a central topic in information theory. The two basic linear functions are defined by the so-called information measures. An *information measure* is an expression $\mu = (\boldsymbol{Y}|\boldsymbol{X})$ or $\sigma = (\boldsymbol{Y};\boldsymbol{Z}|\boldsymbol{X})$, where $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$ are disjoint subsets of the set of variables $\boldsymbol{V}$. We call μ a *monotonicity* and σ a *submodularity* information measure respectively. A monotonicity measure $\mu = (\boldsymbol{Y}|\boldsymbol{X})$ is called *unconditional* iff $\boldsymbol{X} = \emptyset$. Similarly, a submodularity measure $\sigma = (\boldsymbol{Y};\boldsymbol{Z}|\boldsymbol{X})$ is called *unconditional*

2 Given two sets A and B, we use B^A to denote the set of all functions $f : A \to B$.

iff $\boldsymbol{X} = \emptyset$. For any vector $\boldsymbol{h} \in \mathbb{R}_+^{2^V}$, we define the linear functions:

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

We write MON for the set of monotonicity measures, i.e., the set of all $\mu = (\boldsymbol{Y}|\boldsymbol{X})$ where $\boldsymbol{X}, \boldsymbol{Y} \subseteq \boldsymbol{V}$ are disjoint. Similarly, we write SUB for the set of submodularity measures.

A *polymatroid* is a vector $\boldsymbol{h}$ that satisfies the *basic Shannon inequalities*:

$$h(\emptyset) = 0, \qquad \forall \mu \in \mathsf{MON},\ h(\mu) \geq 0, \qquad \forall \sigma \in \mathsf{SUB},\ h(\sigma) \geq 0 \tag{8}$$

The latter two are called monotonicity and submodularity constraints respectively. Every entropic vector is a polymatroid, but the converse is not true [40]. A *Shannon inequality* is a linear inequality that is derived from the basic Shannon inequalities; equivalently, a linear inequality is a Shannon inequality iff it is satisfied by all polymatroids.

Discussion In the literature, the basic Shannon inequalities are often restricted to *elemental* ones, which are submodularity measures of the form $(A; B|\boldsymbol{X})$ and monotonicity measures of the form $(A|\boldsymbol{V} - \{A\})$, where $A, B \in \boldsymbol{V}$ are single variables, and $\boldsymbol{X} \subseteq \boldsymbol{V} - \{A, B\}$. The reason is that the elemental inequalities are sufficient to prove every Shannon inequality; for example $h(B; CD|A) = h(B; C|A) + h(B; D|AC) \geq 0$, when both $h(B; C|A) \geq 0$ and $h(B; D|AC) \geq 0$. Given $m \stackrel{\text{def}}{=} |\boldsymbol{V}|$, the total number of elemental basic Shannon inequalities is $m(m-1)2^{m-3} + m$. [3] However, for the purpose of the PANDA algorithm, it is preferable to consider all basic Shannon inequalities, because this may lead to a smaller exponent of the polylog factor of the algorithm, as we will see in Section 5.

2.5 Statistics on the data

In typical database engines, various statistics about the data are maintained and used for cardinality estimation, query optimization, and other purposes. Common statistics include the number of distinct values in a column, the number of tuples in a relation, and functional dependencies. A robust abstraction called "degree constraints" was introduced in [6] that captures many of these statistics. This section provides a formal definition of degree constraints, which are also the constraints that PANDA can support.

Let $\delta = (\boldsymbol{Y}|\boldsymbol{X})$ be a monotonicity measure, Σ be a database instance with schema Σ, $R \in \Sigma$ be a relation in this instance, and $\boldsymbol{x} \in \mathsf{Dom}^{\boldsymbol{X}}$ be a tuple with schema $\boldsymbol{X}$. We define the quantity *degree* of $\boldsymbol{x}$ with respect to $\boldsymbol{Y}$ in R, denoted by $\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x})$, as follows:

— When both $\boldsymbol{X} \subseteq \mathsf{vars}(R)$ and $\boldsymbol{Y} \subseteq \mathsf{vars}(R)$, then $\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x})$ is the number of times the given $\boldsymbol{X}$-tuple $\boldsymbol{x}$ occurs in $\pi_{\boldsymbol{XY}}(R)$.

3 Specifically, there are $\binom{m}{2}$ ways to choose A and B and 2^{m-2} ways to choose $\boldsymbol{X}$ resulting in $\binom{m}{2} \cdot 2^{m-2}$ elemental submodularities. Additionally, there are m elemental monotonicities.

— When $\boldsymbol{X}$ and $\boldsymbol{Y}$ are arbitrary with respect to $\mathsf{vars}(R)$, then we define the restriction of R to $\boldsymbol{X} \cup \boldsymbol{Y}$ to be the relation $R'(\boldsymbol{XY}) := \mathsf{Dom}^{\boldsymbol{X}\cup\boldsymbol{Y}} \ltimes R$,[4] and set

$$\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) \stackrel{\text{def}}{=} \mathsf{deg}_{R'}(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x})$$

Finally, define the *degree* of the monotonicity measure $\delta = (\boldsymbol{Y}|\boldsymbol{X})$ in R, denoted by $\mathsf{deg}_R(\delta)$, to be

$$\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X}) \stackrel{\text{def}}{=} \max_{\boldsymbol{x}\in\mathsf{Dom}^{\boldsymbol{X}}} \mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X}=\boldsymbol{x}) \tag{9}$$

Note that, for infinite Dom, if $\boldsymbol{Y} \not\subseteq \mathsf{vars}(R)$, then $\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X}) = \infty$, unless $R = \emptyset$, in which case $\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X}) = 0$. We say that R is a *guard* of $\delta = (\boldsymbol{Y}|\boldsymbol{X})$ if $\boldsymbol{Y} \subseteq \mathsf{vars}(R)$, and note that, if $R \neq \emptyset$, then R is a guard of δ iff $\mathsf{deg}_R(\boldsymbol{Y}|\boldsymbol{X}) < \infty$.

If $\boldsymbol{X} = \emptyset$ and $\boldsymbol{Y} = \mathsf{vars}(R)$, then the degree is the *cardinality* of R, $\mathsf{deg}_R(\boldsymbol{Y}|\emptyset) = |R|$. If $\mathsf{deg}_R(\delta) = 1$ then there is a *functional dependency* in R from $\boldsymbol{X}$ to $\boldsymbol{Y}$. If the number of unique values in a column A of R is k, then $\mathsf{deg}_R(A|\emptyset) = k$. Given a schema instance Σ, define the degree of $\delta = (\boldsymbol{Y}|\boldsymbol{X})$ in the instance Σ as:

$$\mathsf{deg}_\Sigma(\delta) \stackrel{\text{def}}{=} \min_{R\in\Sigma} \mathsf{deg}_R(\delta). \tag{10}$$

Let $\Delta \subseteq \mathsf{MON}$ be a set of monotonicity measures and $\boldsymbol{N} : \Delta \to \mathbb{R}_+$ be numerical values for each $\delta \in \Delta$. Throughout this paper, we write N_δ instead of $N(\delta)$, and define $n_\delta \stackrel{\text{def}}{=} \log N_\delta$ for all $\delta \in \Delta$. We view the pair $(\Delta, \boldsymbol{N})$ as a set of *degree constraints*, and we say that an instance Σ *satisfies* the degree constraints $(\Delta, \boldsymbol{N})$ iff $\mathsf{deg}_\Sigma(\delta) \le N_\delta$ for all $\delta \in \Delta$. In that case, we write $\Sigma \models (\Delta, \boldsymbol{N})$.

3. On a Class Information Inequalities

The entropy argument and Shearer's lemma in particular [16] is a powerful information-theoretic tool in extremal combinatorics [33]. Friedgut and Kahn [18] applied the argument to bound the number of homomorphic copies of a graph in a larger graph; this is a special case of the full conjunctive query problem. Grohe and Marx [23], with further elaboration in [9] showed how Shearer's lemma can be used to bound the output size of a full CQ given input cardinality constraints. Briefly, let Σ be the input schema of a full CQ of the form (3),

$$Q(\boldsymbol{V}) \text{ :- } \bigwedge_{R(\boldsymbol{X})\in\Sigma} R(\boldsymbol{X}). \tag{11}$$

where the head atom $Q(\boldsymbol{V})$ has all the variables. Given any non-negative weight vector $\boldsymbol{w} = (w_{\boldsymbol{X}})_{R(\boldsymbol{X})\in\Sigma}$ that forms a fractional edge cover of the hypergraph $(\boldsymbol{V}, \boldsymbol{E})$ where $\boldsymbol{E} := \{\boldsymbol{X} \mid R(\boldsymbol{X}) \in$

4 $S \ltimes T$ denotes the *semi-join reduce* operator defined by $S \ltimes T \stackrel{\text{def}}{=} \pi_{\mathsf{vars}(S)}(S \bowtie T)$.

$\Sigma\}$, the output size of the full CQ is bounded by

$$|Q| \leq \prod_{R(\boldsymbol{X})\in\Sigma} |R|^{w_{\boldsymbol{X}}}. \tag{12}$$

This is known the AGM-bound [10]. The bound is a direct consequence of *Shearer's inequality* [16], which states that the inequality

$$h(\boldsymbol{V}) \leq \sum_{R(\boldsymbol{X})\in\Sigma} w_{\boldsymbol{X}} \cdot h(\boldsymbol{X}), \tag{13}$$

holds for every entropic vector $\boldsymbol{h} \in \mathbb{R}_+^{2^{\boldsymbol{V}}}$ if and only if the weights $\boldsymbol{w}$ form a fractional edge cover of the hypergraph $(\boldsymbol{V}, \boldsymbol{E})$ defined above.

The above results can only deal with cardinality constraints of input relations, and if the input query is a conjunctive query. We extend these results to disjunctive Datalog rules and handle general degree constraints. We start in the next section with a generalization of Shearer's inequality and show how that implies an output cardinality bound for DDRs. In later sections, we use the information inequality to drive the PANDA algorithm.

3.1 Size Bound for DDRs from Information Inequalities

This section develops an information-theoretic bound for the output size of a DDR under general degree constraints. Inequality (14) below is a generalization of Shearer's inequality (13), and the bound (15) is a generalization of the AGM bound (12) for DDRs and general degree constraints. (Recall the notation for the size of a model Σ_{out} in (6).) In what follows, for a given schema Σ and an atom $R(\boldsymbol{X}) \in \Sigma$, for brevity we will also write $\boldsymbol{X} \in \Sigma$ as we assume a one-to-one correspondence between atoms and their variables.

THEOREM 3.1. *Consider a DDR of the form* (5) *with input and output schemas* Σ_{in} *and* Σ_{out} *respectively. Let* $\Delta \subseteq \mathsf{MON}$ *be a set of monotonicity measures. Suppose that there exist two* non-negative *weight vectors* $\boldsymbol{w} := (w_\delta)_{\delta\in\Delta}$ *and* $\boldsymbol{\lambda} := (\lambda_{\boldsymbol{Z}})_{\boldsymbol{Z}\in\Sigma_{\mathsf{out}}}$ *with* $\|\boldsymbol{\lambda}\|_1 = 1$*, where the following inequality holds for all entropic vectors* $\boldsymbol{h}$*:*

$$\sum_{\boldsymbol{Z}\in\Sigma_{\mathsf{out}}} \lambda_{\boldsymbol{Z}} \cdot h(\boldsymbol{Z}) \leq \sum_{\delta\in\Delta} w_\delta \cdot h(\delta) \tag{14}$$

Then, for any input instance Σ_{in} *for the DDR* (5)*, there exists a model* Σ_{out} *for the DDR that satisfies: (Recall that* $|\Sigma_{\mathsf{out}}|$ *is the number of atoms in* Σ_{out}*.)*

$$\|\Sigma_{\mathsf{out}}\| \leq |\Sigma_{\mathsf{out}}| \cdot \prod_{\delta\in\Delta} \left(\mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta)\right)^{w_\delta} \tag{15}$$

PROOF. The plan is to use the entropy argument [16], where we define a uniform probability distribution on a certain subset $\bar{Q} \subseteq \bowtie\Sigma_{\mathsf{in}}$, denote $\boldsymbol{h}$ its entropic vector, then use (14) to prove (15).

Let $\boldsymbol{V} \stackrel{\text{def}}{=} \mathsf{vars}(\Sigma_{\mathsf{in}})$ be the set of all input variables. Notice that, for any joint distribution on $\boldsymbol{V}$ with entropic vector $\boldsymbol{h}$, we have $h(\delta) \leq \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta)$ for all $\delta \in \Delta$. This is trivially true if $\delta \in \Delta$ is not guarded by any relation in Σ_{in} (see Sec 2.5 for the notion of guardedness). Otherwise, the inequality follows from the fact that the uniform distribution on a finite support has the maximum entropy:

$$\begin{aligned} h(\delta) = h(\boldsymbol{Y}|\boldsymbol{X}) &= \sum_{x \in \mathsf{Dom}^{\boldsymbol{X}}} p(\boldsymbol{X} = \boldsymbol{x}) h(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x}) \\ &\leq \sum_{x \in \mathsf{Dom}^{\boldsymbol{X}}} p(\boldsymbol{X} = \boldsymbol{x}) \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x}) \\ &\leq \sum_{x \in \mathsf{Dom}^{\boldsymbol{X}}} p(\boldsymbol{X} = \boldsymbol{x}) \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta) \\ &= \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta) \end{aligned}$$

Next, we construct both the set $\bar{Q}$ and a model Σ_{out} for the DDR as follows. Initially, set $\bar{Q} = \emptyset$, and $Q = \emptyset$ for all $Q \in \Sigma_{\mathsf{out}}$. Iterate over the tuples $\boldsymbol{t} \in \bowtie \Sigma_{\mathsf{in}}$ in some arbitrary order. For each $\boldsymbol{t}$: if $\exists Q \in \Sigma_{\mathsf{out}}$ s.t. $\pi_{\mathsf{vars}(Q)}(\boldsymbol{t}) \in Q$, then ignore $\boldsymbol{t}$; otherwise, insert $\boldsymbol{t}$ into $\bar{Q}$ and insert $\pi_{\mathsf{vars}(Q)}(\boldsymbol{t})$ into Q for every $Q \in \Sigma_{\mathsf{out}}$. In the end, we have constructed a model Σ_{out} for the DDR. Furthermore, $|\bar{Q}| = \|\Sigma_{\mathsf{out}}\|$.

Finally, consider the uniform distribution on $\bar{Q}$, i.e., the distribution where each tuple in Q is chosen randomly with probability $1/|Q|$. Let $\boldsymbol{h}$ be the entropy vector of this distribution. Notice that for each $Q \in \Sigma_{\mathsf{out}}$, the marginal distribution on $\mathsf{vars}(Q)$ is also uniform, because $|Q| = |\bar{Q}|$; in particular, $h(\boldsymbol{Z}) = \log |Q| = \log |\bar{Q}|$ for all $Q(\boldsymbol{Z}) \in \Sigma_{\mathsf{out}}$. Then, noting that $\|\boldsymbol{\lambda}\|_1 = 1$, the following holds:

$$\log \frac{\|\Sigma_{\mathsf{out}}\|}{|\Sigma_{\mathsf{out}}|} = \log |\bar{Q}| = \sum_{\boldsymbol{Z} \in \Sigma_{\mathsf{out}}} \lambda_{\boldsymbol{Z}} \cdot h(\boldsymbol{Z}) \leq \sum_{\delta \in \Delta} w_\delta \cdot h(\delta) \leq \sum_{\delta \in \Delta} w_\delta \cdot \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta)$$

■

In order to obtain the best bound, we need to choose the weights $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ to minimize quantity $\prod_{\delta \in \Delta} \left(\mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta)\right)^{w_\delta}$ on the right-hand side of (15). Specifically, we want to minimize the linear objective

$$\min_{\boldsymbol{\lambda}, \boldsymbol{w}} \sum_{\delta \in \Delta} w_\delta \cdot \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta) \tag{16}$$

subject to the constraints that $\boldsymbol{w} \geq 0$, $\boldsymbol{\lambda} \geq 0$, $\|\boldsymbol{\lambda}\|_1 = 1$, and that inequality (14) holds for all entropic vectors $\boldsymbol{h}$.

For general monotonicity measure Δ, it is an open problem to characterize the weight vectors $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ for which (14) holds for all entropic vectors $\boldsymbol{h}$. In particular, the difficulty is related to the problem of characterizing the entropic region in information theory [38]. Hence,

to make the problem tractable, we relax the upper bound by requiring (14) to hold for all polymatroids $\boldsymbol{h}$.

DEFINITION 3.2 (Polymatroid bound). The following bound is called the *polymatroid bound* for disjunctive datalog rules:

$$b^*_{\Delta,\boldsymbol{N}} \stackrel{\text{def}}{=} \min \sum_{\delta\in\Delta} w_\delta \cdot \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta) \tag{17}$$

$$\begin{aligned} \text{subject to:} \quad & \|\boldsymbol{\lambda}\|_1 = 1 \\ & \text{inequality (14) holds for all polymatroids } \boldsymbol{h} \in \mathbb{R}_+^{2^V} \\ & \boldsymbol{w} \geq 0, \boldsymbol{\lambda} \geq 0 \end{aligned} \tag{18}$$

Note that the bound is stated in log-scale, so that the objective is linear. The constraint (18) is a linear constraint, as explained below. In particular, the polymatroid bound is a linear program.

PROPOSITION 3.3. *The constraint that* (14) *holds for all polymatroids* $\boldsymbol{h}$ *is a linear constraint in the weight vectors* $\boldsymbol{w}$ *and* $\boldsymbol{\lambda}$ *and auxiliary variables.*

PROOF. In the space $\mathbb{R}^{2^V}$, the set of polymatroids is a polyhedron of the form $\{\boldsymbol{h} \mid \boldsymbol{A}\boldsymbol{h} \geq \boldsymbol{0}\}$ for an appropriately defined matrix $\boldsymbol{A}$. Inequality (14) is a linear inequality of the form $\boldsymbol{b}^\top\boldsymbol{h} \geq \boldsymbol{0}$, where $\boldsymbol{b}$ is a linear function of the weights $\boldsymbol{w}$ and $\boldsymbol{\lambda}$. From the Gale-Kuhn-Tucker variant of Farkas' lemma[5], inequality (14) holds for all polymatroids $\boldsymbol{h}$ if and only if there is no $\boldsymbol{h}$ for which $\boldsymbol{b}^\top\boldsymbol{h} < 0$ and $\boldsymbol{A}\boldsymbol{h} \geq \boldsymbol{0}$, which holds if and only if there is a vector $\boldsymbol{x}$ such that $\boldsymbol{A}^\top\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{x} \geq \boldsymbol{0}$. The last condition is a linear constraint in the weights $\boldsymbol{w}$ and $\boldsymbol{\lambda}$, and in the dual variables $\boldsymbol{x}$. ■

The following proposition says that the polymatroid bound linear program always has a rational solution. Therefore, given any degree constraints $(\Delta, \boldsymbol{N})$ and vectors $(\boldsymbol{\lambda}, \boldsymbol{w})$ that define a valid Shannon inequality (14), there are always rational vectors $(\boldsymbol{\lambda}^*, \boldsymbol{w}^*)$ that also define a valid Shannon inequality and satisfy:

$$\sum_{\delta\in\Delta} w^*_\delta \cdot \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta) \leq \sum_{\delta\in\Delta} w_\delta \cdot \log \mathsf{deg}_{\Sigma_{\mathsf{in}}}(\delta)$$

PROPOSITION 3.4. *Given any degree constraints* $(\Delta, \boldsymbol{N})$*, the polymatroid bound linear program from Eq.* (17) *has an optimal solution* $(\boldsymbol{\lambda}^*, \boldsymbol{w}^*)$ *which is rational and independent of* $(\Delta, \boldsymbol{N})$*.*

PROOF. The constraints of the linear program have integer coefficients. Moreover, these coefficients are independent of the given statistics $(\Delta, \boldsymbol{N})$. (The statistics are only used in the

5 https://en.wikipedia.org/wiki/Farkas%27_lemma

objective function.) Consider the polytope defined by the constraints of the linear program. No matter what the objective function is, there is always an optimal solution which is a vertex of this polytope. By Cramer's rule, all these vertices are rational. ■

The question of how tight the polymatroid bound is (and its entropic counterpart) has an intriguing connection to information theory. We refer the reader to [6, 35] for more in-depth discussions and results.

3.2 Equivalent Formulations of Inequality (14)

Shearer's result [16] states that inequality (13) holds for all polymatroids $\boldsymbol{h}$ iff the weights form a fractional edge cover of a certain hypergraph. This section shows an analogous characterization for the generalization (14), and states an "integral" version of this characterization that shall be used by the PANDA algorithm. The following lemma is a variant of Farkas' lemma [34] applied to our specific setting.

LEMMA 3.5. *Let $\boldsymbol{a} \in \mathbb{R}^{2^{\boldsymbol{V}}}$ be a coefficient vector. The following inequality is a Shannon inequality:*

$$\sum_{\boldsymbol{X} \subseteq \boldsymbol{V}} a_{\boldsymbol{X}} h(\boldsymbol{X}) \geq 0 \tag{19}$$

if and of only if there exist non-negative coefficients $\boldsymbol{m} = (m_\mu)_{\mu \in \mathsf{MON}}$ and $\boldsymbol{s} = (s_\sigma)_{\sigma \in \mathsf{SUB}}$ such that the following equality holds as an identity over $2^{|\boldsymbol{V}|}$ symbolic variables $h(\boldsymbol{X})$, $\boldsymbol{X} \subseteq \boldsymbol{V}$:

$$\sum_{\boldsymbol{X} \subseteq \boldsymbol{V}} a_{\boldsymbol{X}} h(\boldsymbol{X}|\emptyset) = \sum_{\mu \in \mathsf{MON}} m_\mu h(\mu) + \sum_{\sigma \in \mathsf{SUB}} s_\sigma h(\sigma) \tag{20}$$

We call the tuple $(\boldsymbol{m}, \boldsymbol{s})$ a witness *of* (19)*. If $\boldsymbol{a}$ is rational, then there exists a rational witness. Furthermore, $\boldsymbol{m}$ and $\boldsymbol{s}$ are a function of $\boldsymbol{a}$ and $|\boldsymbol{V}|$.*

PROOF. Clearly if (20) holds as an identity, then (19) follows trivially. The converse is a direct consequence of a variant of Farkas' lemma, in particular Corollary 7.1h in Schrijver's classic text [34], which states: if the polyhedron $P = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\}$ is not empty, and if inequality $\langle \boldsymbol{c}, \boldsymbol{x} \rangle \geq d$ holds for all $\boldsymbol{x} \in P$, then there exists $d' \geq d$ for which $\langle \boldsymbol{c}, \boldsymbol{x} \rangle \geq d'$ is a non-negative linear combination of the inequalities in $\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$.[6]

In our context, P is the polyhedron defined by (8) and $\boldsymbol{b} = \boldsymbol{0}$, and the inequality is defined by $\boldsymbol{c} = \boldsymbol{a}$ and $d = 0$. Farkas' lemma thus implies that if (19) is a Shannon inequality, then $\langle \boldsymbol{a}, \boldsymbol{h} \rangle \geq 0$ is a non-negative linear combination of the Shannon inequalities in (8), namely there are coefficients $\boldsymbol{m}, \boldsymbol{s}, e^+, e^-$ such that the following identity holds over the variables $h(\boldsymbol{X})$, $\boldsymbol{X} \subseteq \boldsymbol{V}$:

$$\langle \boldsymbol{a}, \boldsymbol{h} \rangle = \sum_{\mu \in \mathsf{MON}} m_\mu h(\mu) + \sum_{\sigma \in \mathsf{SUB}} s_\sigma h(\sigma) + (e^+ - e^-) h(\emptyset). \tag{21}$$

6 We thank the anonymous reviewer for pointing out this specific variant that helps simplify our proof.

where e^+ and e^- are coefficients associated with $h(\emptyset) \geq 0$ and $-h(\emptyset) \geq 0$. Setting $h(\boldsymbol{X}) = 1$ for all $\boldsymbol{X}$ in (21), we conclude that $\langle \boldsymbol{a}, \mathbf{1} \rangle = e^+ - e^-$, and thus (20) holds.

The fact that $\boldsymbol{m}$ and $\boldsymbol{s}$ are a function of $\boldsymbol{a}$ and $|\boldsymbol{V}|$ (and the bounds on their representation sizes) can be found in Chapter 10 of Schrijver's book [34]. ■

In words, Lemma 3.5 says that the LHS of (19) is a positive linear combination of monotonicity and submodularity measures. From the lemma, it is not hard to show that Shearer's inequality (13) holds whenever the weights form a fractional edge cover of the corresponding hypergraph. Given a Shannon inequality of the form (19), the witness $(\boldsymbol{m}, \boldsymbol{s})$ from Lemma 3.5 can be computed by solving the following linear program: The variables are m_μ and s_σ, which are non-negative. The constraints ensure that Eq. (20) is an identity. Specifically, for each $\boldsymbol{X} \subseteq \boldsymbol{V}$, there is a constraint that says that $a_{\boldsymbol{X}}$ must be equal to the weighted sum of terms m_μ and s_σ that contribute to the coefficient of $h(\boldsymbol{X})$ on the RHS of Eq. (20). The linear program has no objective. Instead, we just compute a feasible solution.

To guide the PANDA algorithm later, we will need an "integral" version of the lemma above. Given a set S, a multiset $\mathcal{S}$ *over* S is a multiset whose members are in S. The size of a multiset $\mathcal{S}$, denoted by $|\mathcal{S}|$, is the number of its members, counting multiplicity. If the coefficients $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ in Eq. (14) are rational, then the above linear program has a rational solution $(\boldsymbol{m}, \boldsymbol{s})$, hence the inequality (14) has a rational witness $(\boldsymbol{m}, \boldsymbol{s})$. By multiplying the rational vectors $\boldsymbol{\lambda}$, $\boldsymbol{w}$, $\boldsymbol{m}$, and $\boldsymbol{s}$ with the least common multiple of their denominators, we can convert them to integer vectors. Moreover, the vectors $\boldsymbol{\lambda}$, $\boldsymbol{w}$, $\boldsymbol{m}$, and $\boldsymbol{s}$ are non-negative. We can represent a non-negative integer vector $\boldsymbol{\lambda} = (\lambda_{\boldsymbol{Z}})_{\boldsymbol{Z} \in \Sigma_{\text{out}}}$ as a multiset $\mathcal{Z}$ over Σ_{out} where for every $\boldsymbol{Z} \in \Sigma_{\text{out}}$, the multiplicity of $\boldsymbol{Z}$ in $\mathcal{Z}$ is equal to $\lambda_{\boldsymbol{Z}}$. Similarly, we can represent $\boldsymbol{w}$, $\boldsymbol{m}$, and $\boldsymbol{s}$ as multisets $\mathcal{D}$, $\mathcal{M}$, and $\mathcal{S}$ over Δ, MON, and SUB respectively, leading to the following corollary.

COROLLARY 3.6. *For rational coefficients $\boldsymbol{\lambda}$ and $\boldsymbol{w}$, inequality* (14) *holds for all polymatroids if and only if there exist multisets $\mathcal{Z}$, $\mathcal{D}$, $\mathcal{M}$, and $\mathcal{S}$ over Σ_{out}, Δ, MON, and SUB respectively, such that the following identity holds symbolically over the variables $h(\boldsymbol{X})$, $\boldsymbol{X} \subseteq \boldsymbol{V}$:*

$$\sum_{\boldsymbol{Z} \in \mathcal{Z}} h(\boldsymbol{Z}|\emptyset) = \sum_{\delta \in \mathcal{D}} h(\delta) - \sum_{\mu \in \mathcal{M}} h(\mu) - \sum_{\sigma \in \mathcal{S}} h(\sigma) \tag{22}$$

In particular, if we set $h(\emptyset) = 0$, then the identity (22) *becomes:*

$$\sum_{\boldsymbol{Z} \in \mathcal{Z}} h(\boldsymbol{Z}) = \sum_{\delta \in \mathcal{D}} h(\delta) - \sum_{\mu \in \mathcal{M}} h(\mu) - \sum_{\sigma \in \mathcal{S}} h(\sigma) \tag{23}$$

The sizes of the multisets $\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}$ are bounded by functions of $\boldsymbol{w}$ and $|\boldsymbol{V}|$.

DEFINITION 3.7. We call the terms $h(\delta)$ in (22) and (23) *statistics terms*, and call the terms $h(\mu)$ and $h(\sigma)$ *witness terms*. Specifically, we call terms $h(\mu)$ *monotonicity terms*, and terms $h(\sigma)$ *submodularity terms*. After removing the common denominator in (14), the resulting inequality

is called an *integral Shannon inequality* and has the format:

$$\sum_{\boldsymbol{Z}\in\mathcal{Z}} h(\boldsymbol{Z}) \le \sum_{\delta\in\mathcal{D}} h(\delta) \tag{24}$$

3.3 The Reset Lemma

To conclude this section on information inequalities, we present a combinatorial lemma that plays a key role in our algorithm. The lemma says that, given a valid integral Shannon inequality (24), if we would like to remove an unconditional term $h(\delta_0)$ from the RHS while retaining its validity, then it suffices to remove at most one term $h(\boldsymbol{Z})$ from the LHS. We may be able to remove even more terms from the RHS, in addition to $h(\delta_0)$, but, importantly, it suffices to remove a single term from the LHS.

LEMMA 3.8 (Reset Lemma). *Consider an integral Shannon inequality* (24)*:*

$$\sum_{\boldsymbol{Z}\in\mathcal{Z}} h(\boldsymbol{Z}) \le \sum_{\delta\in\mathcal{D}} h(\delta)$$

Suppose some term $\delta_0 \in \mathcal{D}$ is unconditional*, then there are two multisets $\mathcal{D}' \subseteq \mathcal{D} \setminus \{\delta_0\}$ and $\mathcal{Z}' \subseteq \mathcal{Z}$ with $|\mathcal{Z}'| \ge |\mathcal{Z}| - 1$ such that the following is also an integral Shannon inequality:*

$$\sum_{\boldsymbol{Z}\in\mathcal{Z}'} h(\boldsymbol{Z}) \le \sum_{\delta\in\mathcal{D}'} h(\delta)$$

PROOF. By Corollary 3.6, there exists an integral witness such that equation (23) is an identity (with $h(\emptyset)$ set to 0). As mentioned before, this integral witness can be computed by solving a linear program which gives a rational solution $(\boldsymbol{m}, \boldsymbol{s})$, and then multiplying the rational vectors with the least common multiple of their denominators to convert them to integer vectors, which are then represented as the multisets $\mathcal{M}$ and $\mathcal{S}$ respectively. We prove the lemma by induction on the "potential" quantity $q := |\mathcal{D}| + |\mathcal{M}| + 2|\mathcal{S}|$. The base case when $q \le 1$ is trivial. Consider the case when $q \ge 2$. Suppose $\delta_0 = (\boldsymbol{W} \mid \emptyset)$, i.e., $h(\delta_0) = h(\boldsymbol{W})$.

Since (23) is an identity, the term $h(\boldsymbol{W})$ must cancel out with some other term. If $\boldsymbol{W} \in \mathcal{Z}$, then we can remove $h(\boldsymbol{W})$ from both sides (i.e., setting $\mathcal{Z}' = \mathcal{Z} - \{\boldsymbol{W}\}$ and $\mathcal{D}' = \mathcal{D} - \{(\boldsymbol{W}|\emptyset)\}$). Otherwise, there are three cases:

Case 1 There exists $(\boldsymbol{Y}|\boldsymbol{W}) \in \mathcal{D}$. Then, from

$$h(\boldsymbol{Y}|\boldsymbol{W}) + h(\boldsymbol{W}|\emptyset) = h(\boldsymbol{Y}\boldsymbol{W}|\emptyset) \tag{25}$$

identity (23) remains an identity if we add $(\boldsymbol{Y}\boldsymbol{W}|\emptyset)$ and remove $\{(\boldsymbol{Y}|\boldsymbol{W}), (\boldsymbol{W}|\emptyset)\}$ from $\mathcal{D}$. The potential q decreases by 1, and thus by induction, we can drop the newly added statistics term $h(\boldsymbol{Y}\boldsymbol{W})$ from the RHS of (24) while dropping at most one term from the LHS.

Case 2 $h(\boldsymbol{W})$ cancels with a monotonicity term $h(\mu)$ where $\mu = (\boldsymbol{Y}|\boldsymbol{X})$. In other words, $\boldsymbol{W} = \boldsymbol{XY}$, and the RHS of (23) contains the terms:

$$h(\boldsymbol{W}) - h(\boldsymbol{Y}|\boldsymbol{X}) = h(\boldsymbol{W}) - h(\boldsymbol{XY}) + h(\boldsymbol{X}) = h(\boldsymbol{X}) \tag{26}$$

We add $(\boldsymbol{X}|\emptyset)$ to $\mathcal{D}$, remove $(\boldsymbol{W}|\emptyset)$ from $\mathcal{D}$, and remove $\mu = (\boldsymbol{Y}|\boldsymbol{X})$ from $\mathcal{M}$, thus decreasing the potential q by 1. Then, we proceed inductively to eliminate the newly added statistics term $h(\boldsymbol{X})$.
Case 3 $h(\boldsymbol{W})$ cancels with a submodularity term $h(\sigma)$ where $\sigma = (\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})$ and $\boldsymbol{W} = \boldsymbol{XY}$. In particular, the RHS of (23) contains the terms:

$$\begin{aligned} h(\boldsymbol{W}) - h(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X}) &= h(\boldsymbol{W}) - h(\boldsymbol{XY}) - h(\boldsymbol{XZ}) + h(\boldsymbol{X}) + h(\boldsymbol{XYZ}) \\ &= h(\boldsymbol{XYZ}) - h(\boldsymbol{Z}|\boldsymbol{X}) \end{aligned} \tag{27}$$

We add $(\boldsymbol{XYZ}|\emptyset)$ to $\mathcal{D}$, drop $(\boldsymbol{W}|\emptyset)$ from $\mathcal{D}$, drop $\sigma = (\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})$ from $\mathcal{S}$, and add a new monotonicity measure $(\boldsymbol{Z}|\boldsymbol{X})$ to $\mathcal{M}$. Overall, the potential q decreases by 1. The proof follows by induction where we can eliminate the newly added statistics term $h(\boldsymbol{XYZ})$ from the RHS.

■

We illustrate the reset lemma with the following simple examples:

EXAMPLE 3.9. Consider Shearer's inequality $2h(XYZ) \leq h(XY) + h(YZ) + h(XZ)$, which we write in the form (24) as:[7]

$$h(XYZ) + h(XYZ) \leq h(XY) + h(YZ) + h(XZ) \tag{28}$$

To drop the term $h(XZ)$ on the RHS of (28) while retaining the validity of the inequality, we can also delete one term on the LHS (we can choose any term since they are identical), and obtain the following Shannon inequality:

$$h(XYZ) \leq h(XY) + h(YZ)$$

The proof of the Reset Lemma "explains" how we can do this dropping by reasoning from the identity counterpart of (28) (i.e., an identity of the form (23)):

$$\begin{aligned} h(XYZ) + h(XYZ) = h(XY) + h(YZ) + h(XZ) & \\ - (h(XY) + h(XZ) - h(X) - h(XYZ)) & \quad \text{this is } h(Y; Z|X) \\ - (h(X) + h(YZ) - h(XYZ) - h(\emptyset)) & \quad \text{this is } h(X; YZ|\emptyset) \end{aligned} \tag{29}$$

7 To be consistent with (24), we should write it as $h(XYZ) + h(XYZ) \leq h(XY|\emptyset) + h(YZ|\emptyset) + h(XZ|\emptyset)$.

By canceling out $h(XYZ)$ from both sides, we obtain a different identity:

$$\begin{aligned} h(XYZ) &= h(XY) + h(YZ) \\ &\quad - (h(XY) - h(X)) && \text{this is a monotonicity } h(Y|X) \\ &\quad - (h(X) + h(YZ) - h(XYZ) - h(\emptyset)) && \text{this is } h(X;YZ|\emptyset) \end{aligned} \tag{30}$$

which is what Case 3 from the proof of the Reset Lemma does. The resulting identity witnesses the fact that $h(XYZ) \le h(XY) + h(YZ)$ is a Shannon inequality. ◆

EXAMPLE 3.10. Consider the following Shannon inequality:

$$h(XYZW) + h(Y) \le h(XY) + h(YZ) + h(W|XYZ),$$

which follows from the following identity of the form (23):

$$h(XYZW) + h(Y) = h(XY) + h(YZ) + h(W|XYZ) - h(X;Z|Y)$$

Suppose that we want to drop $h(XY)$ from the RHS. The first step is going to follow Case 3 of the proof of the Reset Lemma by replacing the submodularity $h(X;Z|Y)$ with an additional monotonicity $h(Z|Y)$, thus leading to the identity:

$$h(XYZW) + h(Y) = h(XYZ) + h(YZ) + h(W|XYZ) - h(Z|Y)$$

where our target now is to remove the term $h(XYZ)$ from the RHS. But now, our only option is to use Case 1 and combine $h(XYZ)$ with $h(W|XYZ)$, leading to

$$h(XYZW) + h(Y) = h(XYZW) + h(YZ) - h(Z|Y)$$

And now, we drop $h(XYZW)$ from both sides. The resulting inequality is $h(Y) \le h(YZ)$. ◆

We will also need the following simple proposition to explain a step in PANDA.

PROPOSITION 3.11. *Consider an integral Shannon inequality of the form* (24)*. The number of unconditional statistics terms in $\mathcal{D}$ (counting multiplicities) is at least $|\mathcal{Z}|$.*

PROOF. It is straightforward to verify that the following vector $\boldsymbol{h}$ is a polymatroid, i.e., satisfies (8):

$$h(\boldsymbol{W}) = \begin{cases} 0 & \text{if } \boldsymbol{W} = \emptyset \\ 1 & \text{otherwise} \end{cases}$$

Applying $\boldsymbol{h}$ to (23), the LHS is $|\mathcal{Z}|$, while the RHS is $\le$ to the number of unconditional terms, because $h(\boldsymbol{Y}|\emptyset) = 1$, and $h(\boldsymbol{Y}|\boldsymbol{X}) = 0$ when $\boldsymbol{X} \neq \emptyset$, while all witness terms are non-negative: $h(\mu) \ge 0, h(\sigma) \ge 0$. ■

4. Overview of PANDA and statement of main result

This section states and explains the main result in this paper, which shows that the Shannon inequality (14) is not only useful for bounding the output size of a DDR as shown in Theorem 3.1, but it can also be used to guide an algorithm to evaluate a DDR in time proportional to the bound (15), modulo a polylogarithmic factor. We formally state this result in Section 4.1, and present an illustrative example in Section 4.2. The detailed algorithm description and its proof of correctness are presented in Section 5.

4.1 An Efficient Algorithm to Evaluate Disjunctive Datalog Rules

Given the coefficients $\boldsymbol{w}$ and $\boldsymbol{\lambda}$ of inequality (14) (where recall $\|\boldsymbol{\lambda}\|_1 = 1$), we denote by:

$$B_{\Delta,\boldsymbol{N}} \stackrel{\text{def}}{=} \prod_{\delta\in\Delta} N_\delta^{w_\delta} \qquad\qquad b_{\Delta,\boldsymbol{n}} \stackrel{\text{def}}{=} \log B_{\Delta,\boldsymbol{N}} = \sum_{\delta\in\Delta} w_\delta n_\delta \tag{31}$$

Theorem 3.1 implies that, if Σ_{in} satisfies the degree constraints (see Sec 2.5), i.e., $\Sigma_{\mathsf{in}} \models (\Delta, \boldsymbol{N})$, then there exists a feasible output Σ_{out} that satisfies $\|\Sigma_{\mathsf{out}}\| \le |\Sigma_{\mathsf{out}}| \cdot B_{\Delta,\boldsymbol{N}}$. Our main result states that such an output can be computed in time $\tilde{O}(\|\Sigma_{\mathsf{in}}\| + B_{\Delta,\boldsymbol{N}})$ *if* inequality (14) is a Shannon inequality: (We use $\tilde{O}$ to hide a polylogarithmic factor in the input size $\|\Sigma_{\mathsf{in}}\|$.)

THEOREM 4.1 (The PANDA algorithm for a DDR). *Given the following inputs:*

- *A disjunctive datalog rule (DDR) of the form* (5)*.*
- *An input instance Σ_{in} to the DDR.*
- *Degree constraints $(\Delta, \boldsymbol{N})$ that are satisfied by the instance Σ_{in}, i.e., $\Sigma_{\mathsf{in}} \models (\Delta, \boldsymbol{N})$.*
- *Coefficients $\boldsymbol{w} = (w_\delta)_{\delta\in\Delta}$, and $\boldsymbol{\lambda} = (\lambda_{\boldsymbol{Z}})_{\boldsymbol{Z}\in\Sigma_{\mathsf{out}}}$, with $\|\boldsymbol{\lambda}\|_1 = 1$ that define a valid Shannon inequality* (14)*.*

Let $B_{\Delta,\boldsymbol{N}}$ be the bound defined in (31)*, in terms of the statistics $(\Delta, \boldsymbol{N})$ and the coefficients $\boldsymbol{w}$. Then, we can compute a feasible output Σ_{out} to the DDR in time $\tilde{O}(\|\Sigma_{\mathsf{in}}\| + B_{\Delta,\boldsymbol{N}})$. This feasible output is of size $\tilde{O}(B_{\Delta,\boldsymbol{N}})$.*

We will prove the theorem over the next few sections. We illustrate with a simple example:

EXAMPLE 4.2. Continuing the example in Eq. (7), assume that $|R| = |S| = |U| = N$. Consider the following Shannon inequality, proved by applying two submodularities:

$$h(XY) + h(YZ) + h(ZW) \ge h(XYZ) + h(Y) + h(ZW) \ge h(XYZ) + h(YZW)$$

Rearranging it to the form (14), we obtain:

$$\frac{1}{2}h(XYZ) + \frac{1}{2}h(YZW) \le \frac{1}{2}h(XY) + \frac{1}{2}h(YZ) + \frac{1}{2}h(ZW) \tag{32}$$

Theorem 3.1 proves that there exists a feasible solution (A, B) of size $|A| + |B| \le 2N^{3/2}$, while Theorem 4.1 says that we can compute a feasible solution of size $\tilde{O}(N^{3/2})$ in time $\tilde{O}(N^{3/2})$. ◆

Theorem 4.1 (and PANDA) can be used to compute a full conjunctive query; because a full conjunctive query is a special case of a DDR, where the output schema consists of a single atom containing all variables. To that end, after computing a feasible output of the DDR, we can add this output as an extra atom to the input schema and semijoin-reduce this atom with every other atom in order to obtain a solution to the full conjunctive query. Notice that because this extra atom contains all variables, adding it to the input schema makes the query *acyclic*, hence semijoin reductions are now guaranteed to produce the correct output to the full conjunctive query with no extra tuples [37]. Note also that the output of the full conjunctive query is a *minimal* model to the DDR (Definition 2.2).

COROLLARY 4.3 (The PANDA algorithm for a full conjunctive query). *Suppose that the DDR from Theorem 4.1 corresponds to a full conjunctive query Q, i.e., the output schema of the DDR consists of a single atom containing all variables $\Sigma_{\text{out}} = \{Q(\boldsymbol{V})\}$, in which case the DDR* (5) *collapses back to* (11)*, where $\Sigma := \Sigma_{\text{in}}$. Moreover, suppose that the conditions of Theorem 4.1 are satisfied. Then, the output of the full conjunctive query Q satisfies $|Q| \le B_{\Delta,\boldsymbol{N}}$, and can be computed in time $\tilde{O}(\|\Sigma\| + B_{\Delta,\boldsymbol{N}})$.*

For the best-possible runtime, we want to seed PANDA with parameters minimizing the quantity $B_{\Delta,\boldsymbol{N}}$ (or equivalently, $b_{\Delta,\boldsymbol{N}}$), which is the polymatroid bound (17) for the query Q. In the case of full CQs, there is ony one $\boldsymbol{\lambda}$-parameter, set to $\lambda = 1$, and it remains to find $\boldsymbol{w}$ minimizing $B_{\Delta,\boldsymbol{N}}$. The minimum value is $B^*_{\Delta,\boldsymbol{N}} \stackrel{\text{def}}{=} 2^{b^*_{\Delta,\boldsymbol{N}}}$, defined by

$$\begin{aligned} b^*_{\Delta,\boldsymbol{N}} &\stackrel{\text{def}}{=} \min_{\boldsymbol{w}\ge\boldsymbol{0}} \left\{ \sum_{\delta\in\Delta} w_\delta n_\delta \mid h(\boldsymbol{V}) \le \sum_{\delta\in\Delta} w_\delta h(\delta) \text{ for all polymatroids } \boldsymbol{h} \in \mathbb{R}_+^{2^{\boldsymbol{V}}} \right\} \\ &= \max_{\boldsymbol{h}\in\mathbb{R}_+^{2^{\boldsymbol{V}}}} \left\{ h(\boldsymbol{V}) \mid h(\delta) \le n_\delta \text{ for all } \delta \in \Delta \text{ and } \boldsymbol{h} \text{ is a polymatroid} \right\} \end{aligned}$$

The second equality follows from simple duality arguments. PANDA would be worst-case optimal if $B^*_{\Delta,\boldsymbol{N}}$ is tight, in the sense that we can construct a database instance Σ that satisfies the degree constraints $(\Delta, \boldsymbol{N})$ and has output size $|Q| = \Omega(B^*_{\Delta,\boldsymbol{N}})$. There are two special cases where $B^*_{\Delta,\boldsymbol{N}}$ is known to be tight in data complexity: when all degree constraints are "simple" (see [2, 35]), or when the degree constraints are "acyclic" (see [5, 31]). The degree constraints are *simple* if $|\boldsymbol{X}| \le 1$ for all $\delta = (\boldsymbol{Y}|\boldsymbol{X}) \in \Delta$, and they are *acyclic* if there is a global order of the variables in $\boldsymbol{V}$ such that for every $\delta = (\boldsymbol{Y}|\boldsymbol{X}) \in \Delta$, all variables in $\boldsymbol{X}$ precede all variables in $\boldsymbol{Y}$ in the order. If Δ contains only cardinality constraints, then it is both simple and acyclic, and $B^*_{\Delta,\boldsymbol{N}}$ is exactly the AGM bound because (14) reduces back to (13).

4.2 Example: Preview of PANDA

We illustrate the algorithm with the DDR (7):

$$A(X,Y,Z) \vee B(Y,Z,W) \text{ :- } R(X,Y) \wedge S(Y,Z) \wedge U(Z,W)$$

Following Example 4.2, we assume we only have cardinality statistics / constraints:

$$\Delta = \{(XY|\emptyset), (YZ|\emptyset), (ZW|\emptyset)\}, \qquad N_{XY} = N_{YZ} = N_{ZW} = N.$$

We further assume that N is a power of 2. Each constraint has a "guard", i.e., a relation that "sponsors" (satisfies) the constraint: $T_{XY} := R, T_{YZ} := S, T_{ZW} := U$. We name all guards $T_\cdot$ to be consistent with the formal description of the algorithm presented in Section 5.

Suppose that the input Shannon inequality given is the one in Eq (32). This inequality is in the shape of (14) with $\lambda_{XYZ} = \lambda_{YZW} = 1/2$ and $w_{XY|\emptyset} = w_{YZ|\emptyset} = w_{ZW|\emptyset} = 1/2$. The fact that this is a Shannon inequality was shown above in Example 4.2.

From Corollary 3.6, the Shannon inequality (32) must have a corresponding identity (23) over symbolic variables $h(\boldsymbol{X})$ where $h(\emptyset) = 0$. We show one such identity below:[8]

$$h(XYZ) + h(YZW) = h(XY|\emptyset) + h(YZ|\emptyset) + h(ZW|\emptyset) - h(X;Z|Y) - h(Y;ZW|\emptyset) \tag{33}$$

The bound from (31) is $b_{\Delta,N} = (n_{XY} + n_{YZ} + n_{ZW})/2 = 3n/2$ where $n \overset{\text{def}}{=} \log N$, or equivalently, $B_{\Delta,N} = N^{3/2}$; this is the runtime budget the algorithm cannot exceed, in order to compute a feasible solution (A, B) to the DDR.

Our algorithm operates by observing and manipulating identity (33), while maintaining its identity invariant. Every step of the algorithm applies a modification to the identity and mirrors this modification with an algorithmic computation to create one or more sub-problems. Each sub-problem is then equipped with the newly created identity, the newly created input data, and carry on the next step on their own. The spawned sub-problems form a (sub-problem) tree. In the end, we gather results from the leaves of this tree to answer the original query.

In identity (33), the statistics terms $h(XY|\emptyset)$, $h(YZ|\emptyset)$, and $h(ZW|\emptyset)$ correspond to input data (input relations) R, S, and U. When we modify the identity, we will be transforming some subsets of these statistics terms into different statistics terms; correspondingly, the input relations are manipulated to create new input relations that are associated with the new statistics terms. We next describe this idea in more details. The steps of the algorithm are also shown in Figure 1.

In the first step, we grab an unconditional statistics term (from $\mathcal{D}$) in (33). Let's say the term we grab is $h(XY|\emptyset) = h(XY)$. Since (33) is an identity, there must be another term that cancels out the symbolic variable $h(XY)$. In this case, it is $-h(X;Z|Y)$; so we combines $h(XY)$

8 Our witness is not elemental, because $h(Y;ZW|\emptyset)$ is not elemental: if we replaced the latter with $h(Y;W|\emptyset) + h(Y;Z|W)$, then we obtain an elemental witness. Also note that the witness is not unique; for example, we could have used the witness $h(XY;Z|\emptyset) + h(Y;W|Z)$.

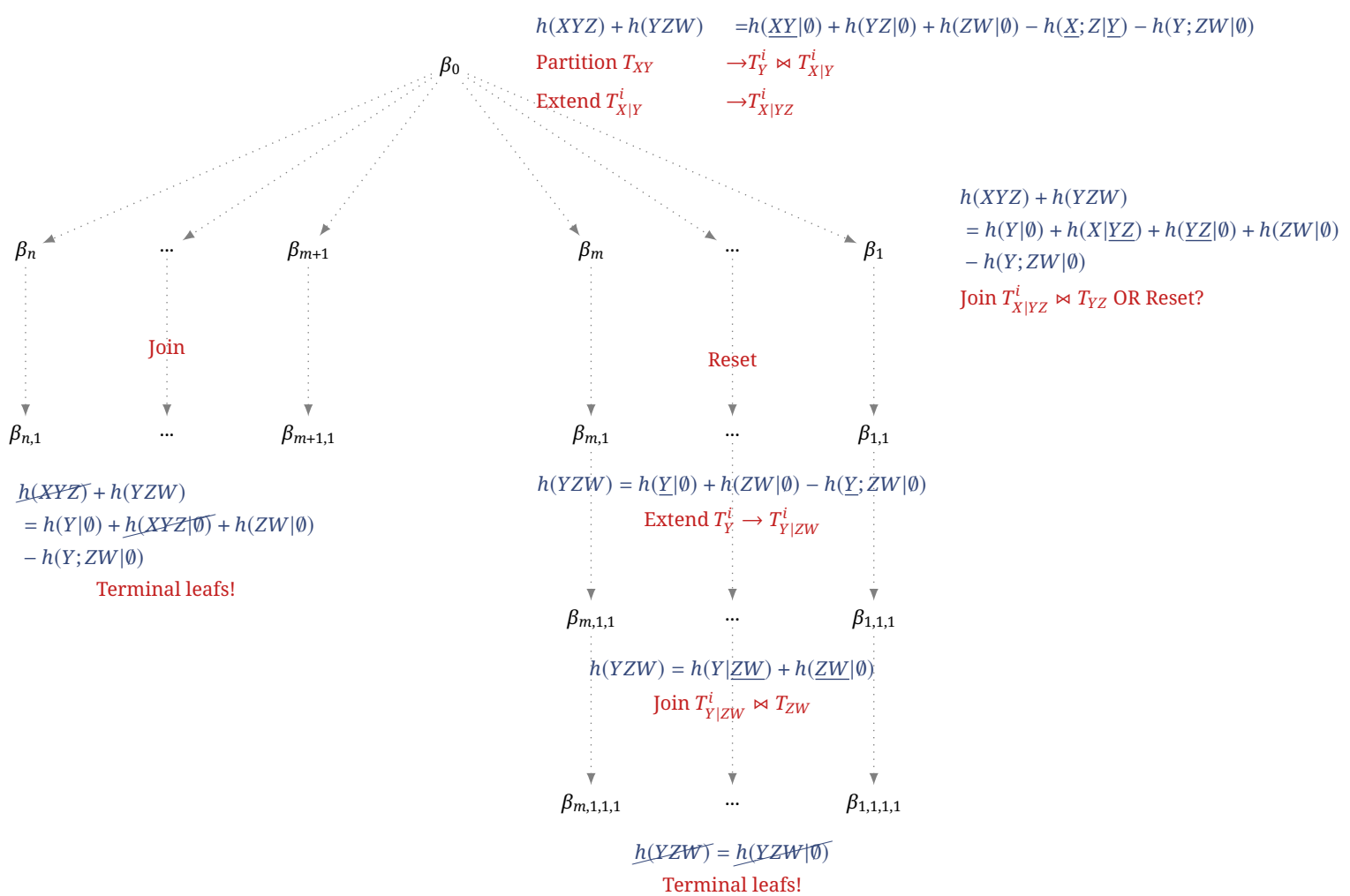


Figure 1. An example illustrating the PANDA algorithm over the disjunctive rule (7). Here, $n \stackrel{\text{def}}{=} \log N$ and $m \stackrel{\text{def}}{=} \frac{n}{2}$. For each node in the sub-problem tree, the corresponding identity (23) is in blue while the corresponding algorithmic operation is in red.

with the canceling term to obtain new statistics terms:

$$\begin{aligned} h(\underline{XY}|\emptyset) - h(\underline{X}; Z|\underline{Y}) &= h(XY) - h(\emptyset) - h(XY) - h(YZ) + h(XYZ) + h(Y) \\ &= h(Y) - h(\emptyset) + h(XYZ) - h(YZ) \\ &= h(Y|\emptyset) + h(X|YZ) \end{aligned}$$

This rewriting-by-cancellation changed our identity (33) into a new one:

$$h(XYZ) + h(YZW) = h(Y|\emptyset) + h(X|YZ) + h(YZ|\emptyset) + h(ZW|\emptyset) - h(Y; ZW|\emptyset) \tag{34}$$

The change amounts to replacing a statistics term with two new ones: $h(XY|\emptyset) \rightarrow h(Y|\emptyset) + h(X|YZ)$. We mimic this symbolic change with an actual relational change that shall alter the structure of the problem. The rough idea is to create inputs to the new problem by creating two new guards for the two new statistics terms:

- The guard for $h(Y|\emptyset)$ will be the table $T_Y \stackrel{\text{def}}{=} \pi_Y(R)$.
- The guard for $h(X|YZ)$ is obtained by using $R(X, Y)$ to construct a "dictionary" (a lookup table) which supports the following operation: given a tuple (y, z), return the list of all

x-values such that $(x, y) \in R$. (Note that this operation does not use the given z-value.) We denote this dictionary as $T_{X|YZ}$.

The aim is for us to be able to join all 4 guards (corresponding to the four statistics terms on the RHS of (34)) to answer the same query as before. However, implementing this idea straight up does not work, because the new degree constraint for T_Y is $N_Y \stackrel{\text{def}}{=} |\pi_Y(R)|$ and for $T_{X|YZ}$ is $N_{X|YZ} \stackrel{\text{def}}{=} \deg_R(X|Y)$, and they are too large: the product $N_Y \cdot N_{X|YZ}$ can be much greater than the original statistics of $N_{XY} = N = |R|$ (guarding $h(XY)$ that we started with). If this product is too large, we cannot apply induction to solve the sub-problem in our time budget of $B_{\Delta,\boldsymbol{N}} = N^{3/2}$.

The reason this product $N_Y \cdot N_{X|YZ}$ is too large is that N_Y counts both high-degree and low-degree y-values, while the new statistics $N_{X|YZ}$ is the maximum degree. Thus, the product is an over-estimation of what the size of the join $T_Y \bowtie T_{X|YZ}$ can be. To remedy the situation, we *uniformize* the problem by partitioning R into a logarithmic number of sub-relations $R = R^1 \cup \cdots \cup R^k$, where each sub-relation contains tuples whose y-values have similar degrees. In effect, uniformization is an algorithmic technique for dealing with *skews*, a notoriously well-known reason for why query plans might blow up in practice.[9]

Concretely, the partitioning is done as follows. Relation R^i contains all tuples $(x, y) \in R$ where:

$$\frac{N}{2^i} \leq \deg_R(X|Y = y) < \frac{N}{2^{i-1}}$$

We set $T_Y^i \stackrel{\text{def}}{=} \pi_Y(R^i)$, thus $|T_Y^i| \leq 2^i$. We further partition T_Y^i into two equal-sized buckets (which we will continue to name "buckets i", with some abuse). The two new guards T_Y^i and R^i have statistics N_Y^i and $N_{X|YZ}^i$, which (by our partition-into-two trick) satisfy the following:

$$N_Y^i \stackrel{\text{def}}{=} |T_Y^i| \leq 2^{i-1}$$
$$N_{X|YZ}^i \stackrel{\text{def}}{=} \deg_{R^i}(X|Y) \leq \frac{N}{2^{i-1}}$$
$$N_Y^i \cdot N_{X|YZ}^i \leq N$$

To be consistent with the notation used in describing PANDA, all guards will be called T: in particular, the two new guards $\pi_Y(R^i)$ and R^i are called T_Y^i and $T_{X|YZ}^i$ respectively.

---

9 To briefly explain this insight, consider the triangle query $Q(x, y, z)$:- $R(x, y) \wedge S(y, z) \wedge T(x, z)$. It is straightforward to construct input instances for which $|R| = |S| = |T| = N$, and $|R \bowtie S|$, $|S \bowtie T|$, and $|R \bowtie T|$ are all $\Omega(N^2)$, by having high-degree (i.e., skewed) values to join [**32**]. To overcome this limitation of join-project query plans, one variant of worst-case optimal join algorithms does the following. Partition $R = R^h \cup R^\ell$, where $R^h \stackrel{\text{def}}{=} \{(x, y) \in R \mid \deg_R(Y|X = x) > \sqrt{N}\}$ and $R^\ell \stackrel{\text{def}}{=} R \setminus R^h$.
The query can be rewritten as $Q(x, y, z)$:- $R^h(x, y) \wedge S(y, z) \wedge T(x, z) \vee R^\ell(x, y) \wedge S(y, z) \wedge T(x, z)$. The joins $R^h(x, y) \wedge S(y, z)$ and $R^\ell(x, y) \wedge T(x, z)$ are both cardinality-bounded by $N^{3/2}$ and computable within $O(N^{3/2})$-time. Thus, the entire query can be computed within $O(N^{3/2})$-time.
Our uniformization step is a generalization of this idea to the setting of DDRs. We needed to partition R into k parts instead of 2 parts to maintain our algorithmic invariants. For some special classes of queries, Bringmann and Gorbachev [**14**] show that heavy/light partitioning is sufficient to achieve the same result.

After partitioning, we now have $O(\log N)$ sub-problems, each equipped with identity (34). In the second step, we again grab an unconditional statistics term $h(YZ|\emptyset)$, and find a term from (34) to cancel it.[10] This time, the cancellation is:

$$h(X|\underline{YZ}) + h(\underline{YZ}|\emptyset) = h(XYZ) - h(YZ) + h(YZ) - h(\emptyset) = h(XYZ|\emptyset),$$

leading to a new identity

$$h(XYZ) + h(YZW) = h(Y|\emptyset) + h(XYZ|\emptyset) + h(ZW|\emptyset) - h(Y;ZW|\emptyset) \tag{35}$$

The change $h(X|\underline{YZ}) + h(\underline{YZ}|\emptyset) \to h(XYZ)$ is suggestive of a join, where for the ith subproblem, we will join the corresponding guards : $T^i_{X|YZ} \bowtie T_{YZ}$. Recall that $N^i_{X|YZ} \le N/2^{i-1}$, hence for $i > \frac{1}{2}\log N$ performing this join won't take time over the budget of $O(N^{3/2})$. On the other hand, when $i \le \frac{1}{2}\log N$, we will need an additional idea, to use the reset lemma (Lemma 3.8) to reroute the sub-problem away from the "heavy-hitter hotspot", i.e., join over the high-degree Y-values. In addition to uniformization, the reset lemma is our second ingredient to deal with skews.

Concretely, for the ith sub-problem, we do the following:

— When $i > \frac{1}{2}\log N$, then PANDA performs a join $T^i_{XYZ} := T^i_{X|YZ} \bowtie T_{YZ}$. The output size of this join is within the bound $B_{\Delta,\boldsymbol{N}} = N^{3/2}$. After computing this join, there will be no more sub-problems because we have computed a relation that fits the output schema, namely it corresponds to $A(X,Y,Z)$. We add tuples from T^i_{XYZ} to the output relation $A(X,Y,Z)$.

— In the case where $i \le \frac{1}{2}\log N$, the algorithm attempts to perform the same join $T^i_{XYZ} := T^i_{X|YZ} \bowtie T_{YZ}$, but its output size now exceeds the bound $B_{\Delta,\boldsymbol{N}} = N^{3/2}$. Therefore, the algorithm does not compute a guard T^i_{XYZ} for $h(XYZ|\emptyset)$, but instead uses the reset lemma to cancel out this term with the term $h(XYZ|\emptyset)$ on the LHS. The new identity (for these sub-problems) is now

$$h(YZW) = h(Y|\emptyset) + h(ZW|\emptyset) - h(Y;ZW|\emptyset)$$

We again grab an unconditional statistics term $h(Y|\emptyset)$ and cancel it with $h(\underline{Y};ZW|\emptyset)$:

$$h(Y|\emptyset) - h(Y;ZW|\emptyset) = h(Y|ZW)$$

The guard T^i_Y for $h(Y|\emptyset)$ has size $|T^i_Y| \le N^{1/2}$, thanks to the fact that $|T^i_Y| \le 2^{i-1}$ and $i \le \frac{1}{2}\log N$, The above step replaces $h(Y|\emptyset)$ with a new statistics term $h(Y|ZW)$. Its guard is computed from T^i_Y, by extending it into a dictionary $T^i_{Y|ZW}$: given (z,w), this dictionary returns all y-values for which $(y) \in T^i_Y$. In particular, this silly dictionary always returns the entire table T^i_Y, no matter what the given (z,w) are.[11] After the dictionary extension, the algorithm performs a join $T^i_{YZW} := T^i_{Y|ZW} \bowtie T_{ZW}$. This join is feasible since the output

10 Note how "greedy" the algorithm is. This is one of the reasons for the large polylog factor it suffers.

11 To be more precise, before PANDA extends T^i_Y into $T^i_{Y|ZW}$, it will "uniformize" T^i_Y by partitioning the table T^i_Y into $\log|T^i_Y|$ buckets based on the "degree" $\deg_{T^i_Y}(Y|\emptyset)$. However, this partition is vacuous since only one bucket will be non-empty.

size is within the bound $B_{\Delta,\boldsymbol{N}} = N^{3/2}$. Now, the algorithm reaches a terminal node since the join result T^i_{YZW} is in the output schema; namely, it corresponds to $B(Y,Z,W)$.

At the end of the algorithm, the tables T^i_{XYZ} from all branches are unioned into the output relation $A(X,Y,Z)$, while tables T^i_{YZW} from all branches are unioned into the other output relation $B(Y,Z,W)$. These two relations are the final output of the algorithm for the DDR (7). Note that the input table T_{ZW} does not contribute to the output table $A(X,Y,Z)$, and similarly the input table T_{YZ} does not contribute to the output table $B(Y,Z,W)$. Nevertheless, the tables $A(X,Y,Z)$ and $B(Y,Z,W)$ are still a valid output to the DDR. This is because Definition 2.2 only requires that if a tuple (x,y,z,w) satisfies the conjunction $R(x,y) \wedge S(y,z) \wedge U(z,w)$, then it must satisfy the disjunction $A(x,y,z) \vee B(y,z,w)$. We do not require the converse to hold in DDR semantics. (The converse is only required in CQ semantics, as we will see in Section 6.)

Note that, for the particular query (7), there is a way to compute a feasible output without the polylog factor as described above. The above strategy is only meant to illustrate the PANDA algorithm in its full generality.

5. Detailed Description of PANDA

This section describes the PANDA algorithm in detail. Section 5.1 presents the main data structures (called tables and dictionaries) used by the algorithm. Section 5.2 presents the algorithm and its proof of correctness and runtime analysis.

5.1 Tables and Dictionaries

For a given statistics term $\delta = (\boldsymbol{Y}|\boldsymbol{X})$, PANDA uses two kinds of data structures: *tables* and *dictionaries*, denoted T_δ. When $\boldsymbol{X} = \emptyset$, then we call it a *table*; otherwise, we call it a *dictionary*. There will be at most one table/dictionary T_δ for a given δ. As usual, we abbreviate $\boldsymbol{Y}|\emptyset$ with just $\boldsymbol{Y}$, and statistics $N_{\boldsymbol{Y}|\emptyset}, n_{\boldsymbol{Y}|\emptyset}$ with just $N_{\boldsymbol{Y}}$ and $n_{\boldsymbol{Y}}$. Specifically, a table is a set $T_{\boldsymbol{Y}} \subseteq \mathsf{Dom}^{\boldsymbol{Y}}$ of tuples over the $\boldsymbol{Y}$ variables, and a dictionary is a function $T_{\boldsymbol{Y}|\boldsymbol{X}} : \mathsf{Dom}^{\boldsymbol{X}} \to 2^{\mathsf{Dom}^{\boldsymbol{Y}}}$ that gives a set of tuples over the $\boldsymbol{Y}$ variables *given* a specific binding $\boldsymbol{X} = \boldsymbol{x}$ of the $\boldsymbol{X}$-variables.

For a statistics term $\delta = (\boldsymbol{Y}|\boldsymbol{X})$, each table/dictionary T_δ is associated with a *statistics* N_δ. We say that T_δ *satisfies* the statistics, and we write $T_\delta \models N_\delta$, iff $|T_\delta(\boldsymbol{x})| \le N_\delta$ for all $\boldsymbol{x} \in \mathsf{Dom}^{\boldsymbol{X}}$. As a special case, a table $T_{\boldsymbol{Y}}$ *satisfies* a statistics $N_{\boldsymbol{Y}}$ iff $|T_{\boldsymbol{Y}}| \le N_{\boldsymbol{Y}}$.

The algorithm performs the following operations on tables and dictionaries: join, projection, extension, construction, and partition. Each operation yields the corresponding statistics on the results, as described below.

— *Join* of a table with a dictionary, $T_{\boldsymbol{XY}} := T_{\boldsymbol{X}} \bowtie T_{\boldsymbol{Y}|\boldsymbol{X}}$. This operation constructs a new table with statistics $N_{\boldsymbol{XY}} \stackrel{\text{def}}{=} N_{\boldsymbol{X}} N_{\boldsymbol{Y}|\boldsymbol{X}}$. The join takes time $O(N_{\boldsymbol{X}} N_{\boldsymbol{Y}|\boldsymbol{X}})$.

— *Projection* of a table, $T_{\boldsymbol{X}} := \pi_{\boldsymbol{X}}(T_{\boldsymbol{XY}})$. This operation takes a table $T_{\boldsymbol{XY}}$ over variables $\boldsymbol{X} \cup \boldsymbol{Y}$, with statistics $N_{\boldsymbol{XY}}$, and constructs a new table $T_{\boldsymbol{X}}$ with statistics $N_{\boldsymbol{X}} \stackrel{\text{def}}{=} N_{\boldsymbol{XY}}$. The projection takes time $O(N_{\boldsymbol{XY}})$.

— *Extension* of a dictionary $T_{\boldsymbol{Y}|\boldsymbol{X}}$ into another dictionary $T_{\boldsymbol{Y}|\boldsymbol{XZ}}$, where $\boldsymbol{Z}$ is disjoint from $\boldsymbol{XY}$. This operation takes as input a dictionary $T_{\boldsymbol{Y}|\boldsymbol{X}}$ and returns a new dictionary $T_{\boldsymbol{Y}|\boldsymbol{XZ}}$ defined as $T_{\boldsymbol{Y}|\boldsymbol{XZ}}(\boldsymbol{x}, \boldsymbol{z}) \stackrel{\text{def}}{=} T_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{x})$ for all $(\boldsymbol{x}, \boldsymbol{z}) \in \mathsf{Dom}^{\boldsymbol{XZ}}$. Its statistics is $N_{\boldsymbol{Y}|\boldsymbol{XZ}} \stackrel{\text{def}}{=} N_{\boldsymbol{Y}|\boldsymbol{X}}$. This operation takes $O(1)$ time, because the operation does not touch the data, but only constructs a new function that calls the existing function $T_{\boldsymbol{Y}|\boldsymbol{X}}$.

— *Construction.* Given a table $T_{\boldsymbol{XY}}$ over variables $\boldsymbol{X} \cup \boldsymbol{Y}$ with statistics $N_{\boldsymbol{XY}}$, construct a dictionary $T_{\boldsymbol{Y}|\boldsymbol{X}}$, with statistics $N_{\boldsymbol{Y}|\boldsymbol{X}} \stackrel{\text{def}}{=} \mathsf{deg}_{T_{\boldsymbol{XY}}}(\boldsymbol{Y}|\boldsymbol{X})$. This operation takes $O(N_{\boldsymbol{XY}})$ time because it involves scanning the table $T_{\boldsymbol{XY}}$ and constructing an index. The operation returns a function that looks up in this index.

— *Partition.* Given a table $T_{\boldsymbol{XY}}$ over variables $\boldsymbol{X} \cup \boldsymbol{Y}$ with statistics $N_{\boldsymbol{XY}}$, partition $T_{\boldsymbol{XY}}$ into $k := 2^{\lceil \log |T_{\boldsymbol{XY}}| \rceil}$ many sub-tables $T^1, \ldots, T^k$ satisfying the conditions stated in Lemma 5.1 below. This operation takes time $O(N_{\boldsymbol{XY}})$.

LEMMA 5.1. *Let $T_{\boldsymbol{XY}}$ be a table over variables $\boldsymbol{X} \cup \boldsymbol{Y}$ with statistics $N_{\boldsymbol{XY}}$. Then, $T_{\boldsymbol{XY}}$ can be partitioned into at most $k := 2^{\lceil \log |T_{\boldsymbol{XY}}| \rceil}$ sub-tables $T^1, \ldots, T^k$ satisfying*

$$|\pi_{\boldsymbol{X}}(T^i)| \cdot \deg_{T^i}(\boldsymbol{Y}|\boldsymbol{X}) \le N_{\boldsymbol{XY}}, \qquad \forall i \in [k]. \tag{36}$$

PROOF. To obtain the sub-tables T^i, observe that the number of tuples $\boldsymbol{x} \in \pi_{\boldsymbol{X}}(T_{\boldsymbol{XY}})$ with log-degree in the interval $[i, i+1)$ is at most $|T_{\boldsymbol{XY}}|/2^i \le 2^{n_{\boldsymbol{XY}} - i}$. Hence, if we partition $T_{\boldsymbol{XY}}$ based on which of the buckets $[i, i+1)$ the log-degree falls into, we will almost attain the required inequality, just off by a factor of 2:

$$|\pi_{\boldsymbol{X}}(T^i)| \cdot \deg_{T^i}(\boldsymbol{Y}|\boldsymbol{X}) \le 2^{(n_{\boldsymbol{XY}} - i)} \cdot 2^{(i+1)} = 2N_{\boldsymbol{XY}}.$$

To get rid of the factor of 2, we partition each T^i into two tables whose projections onto $\boldsymbol{X}$ are equally sized. Overall, we need $k := 2^{\lceil \log |T_{\boldsymbol{XY}}| \rceil}$ partitions. ∎

5.2 Algorithm

This section describes PANDA and proves Theorem 4.1. The main input to PANDA contains an input instance Σ_{in}, an output schema $\Sigma_{\mathsf{out}} \neq \emptyset$, and statistics $(\Delta, \boldsymbol{N})$ satisfied by the input instance, i.e., $\Sigma_{\mathsf{in}} \models (\Delta, \boldsymbol{N})$ as defined in Section 2.5. In addition, we are also given the coefficients $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ for which (14) is a Shannon inequality. By Proposition 3.4, we can assume that $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ are rational and independent of the statistics $(\Delta, \boldsymbol{N})$. From Corollary 3.6, we can also assume that PANDA was given the multisets $\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}$ for which identity (23) holds. In particular, as

in the proof of Corollary 3.6, given vectors $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ that make Eq. (14) a Shannon inequality, the witness $(\boldsymbol{m}, \boldsymbol{s})$ can be computed by solving a linear program. Since $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ are rational and independent of the data, the solution $(\boldsymbol{m}, \boldsymbol{s})$ is also rational and independent of the data. By multiplying the vectors $\boldsymbol{\lambda}$, $\boldsymbol{w}$, $\boldsymbol{m}$, and $\boldsymbol{s}$ with the least common multiple of their denominators, we can convert them to integer vectors. And now we can represent the integral $\boldsymbol{\lambda}, \boldsymbol{w}, \boldsymbol{m}$ and $\boldsymbol{s}$ as multisets $\mathcal{Z}$, $\mathcal{D}$, $\mathcal{M}$, and $\mathcal{S}$ respectively. All the above steps take constant time in *data complexity*; see Section 2.1. This is because the vectors $\boldsymbol{\lambda}$, $\boldsymbol{w}$, $\boldsymbol{m}$, and $\boldsymbol{s}$ have dimension $O(1)$ in data complexity. Moreover, since the values of these vectors don't depend on the data, they are considered constants in data complexity.

We shall show that a feasible output Σ_{out} to the DDR (5) can be computed in time $\tilde{O}(\|\Sigma_{\mathsf{in}}\| + B_{\Delta,\boldsymbol{N}})$, defined in (31). In terms of these multisets, the bounds defined in (31) have the following equivalent expressions:

$$B_{\Delta,\boldsymbol{N}} = \left(\prod_{\delta\in\mathcal{D}} N_\delta\right)^{1/|\mathcal{Z}|} \qquad b_{\Delta,\boldsymbol{n}} = \log B_{\Delta,\boldsymbol{N}} = \frac{1}{|\mathcal{Z}|}\sum_{\delta\in\mathcal{D}} n_\delta \tag{37}$$

For each statistics term $\delta = (\boldsymbol{Y}|\boldsymbol{X}) \in \mathcal{D}$, there is a guard which is a table/dictionary instance $T_{\boldsymbol{Y}|\boldsymbol{X}}$ with statistics $N_{\boldsymbol{Y}|\boldsymbol{X}}$, i.e., $T_{\boldsymbol{Y}|\boldsymbol{X}} \models N_{\boldsymbol{Y}|\boldsymbol{X}}$ as defined in Section 5.1. Creating the initial guards can be done in time $\tilde{O}(\|\Sigma_{\mathsf{in}}\|)$. We adopt the convention that the lower case letters $n_\delta, b_{\Delta,\boldsymbol{n}}$ represent the logarithms of the upper case $N_\delta, B_{\Delta,\boldsymbol{N}}$ respectively.

Summarized in Algorithm 1, PANDA works as follows. Starting from an initial node, it grows a tree of sub-problems. New sub-problems are spawned from a "non-terminal" leaf node of the tree. The process stops when every leaf node is terminal, a concept we shall define shortly. After the tree growing stops, a feasible output of the problem is gathered from the leaves of the tree.

To every node, there associates a sub-problem parameterized by a tuple $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$ where

- The multisets $\mathcal{Z}, \mathcal{D}, \mathcal{M}$, and $\mathcal{S}$ are over the base sets Σ_{out}, Δ, MON, and SUB, respectively. These multisets will maintain the invariant that identity (23) holds, as we shall show in the next section.
- $\boldsymbol{n}$ is a collection of non-negative real numbers (logs of statistics), one statistics n_δ for each $\delta \in \mathcal{D}$. (Recall that, $n_\delta := \log N_\delta$, for convenience.)
- $\boldsymbol{T}$ is a collection of dictionaries, one dictionary T_δ for each $\delta \in \mathcal{D}$; these shall be the guards for $N_\delta = 2^{n_\delta}$. In particular, $T_\delta \models N_\delta$ for each $\delta \in \mathcal{D}$.

DEFINITION 5.2 (Terminal leaf). A leaf node of the tree is "terminal" (it won't spawn off new sub-problem(s) anymore) if its parameters $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$ are such that, there is an unconditional statistics term $(\boldsymbol{Z}|\emptyset) \in \mathcal{D}$ for which $\boldsymbol{Z} \in \mathcal{Z}$.

We next briefly explain in words the key steps of Algorithm 1. A proof that the algorithm is correct and an analysis of its runtime to show Theorem 4.1 are described in Section 5.3. The main loop looks for a non-terminal leaf node β of the sub-problem tree. If there is no such leaf node, then the code-block starting at line 31 gathers the results from the parameters of the (terminal) leaf nodes to construct the final output Σ_{out}.

If there is a non-terminal leaf node β, then we pick an arbitrary unconditional[12] statistics term $\delta = (\boldsymbol{W}|\emptyset) \in \mathcal{D}$ (line 4) and start considering cases in parallel with the cases in the proof of Lemma 3.8, except that Case 3 will be handled differently. The major insight of our work is that we can design an algorithm where the algorithmic steps mirror the Shannon inequality inference steps:

Case 1: Join or Reset There exists a statistics term $(\boldsymbol{Y}|\boldsymbol{W}) \in \mathcal{D}$ (line 5). If the statistics are such that the join-size of $T_{\boldsymbol{W}} \bowtie T_{\boldsymbol{Y}|\boldsymbol{W}}$ is smaller than the budget $B_{\Delta,N}$, checked at line 6, then we compute the join result $T_{\boldsymbol{YW}}$ as shown, and let it guard the new statistics term $(\boldsymbol{YW}|\emptyset)$. This case corresponds precisely to applying Eq. (25), and the new multiset $\mathcal{D}'$ defined in line 8 reflects that. After joining $T_{\boldsymbol{W}}$ and $T_{\boldsymbol{Y}|\boldsymbol{W}}$ to form $T_{\boldsymbol{YW}}$, we remove the old dictionaries from $\boldsymbol{T}$ and add the new dictionary to $\boldsymbol{T}$. Similarly, the statistics in $\boldsymbol{n}$ are replaced in the same way. On the other hand, if the join $T_{\boldsymbol{W}} \bowtie T_{\boldsymbol{Y}|\boldsymbol{W}}$ is larger than the budget, then we perform a *reset* step. The Shannon inequality reasoning is as follows. Initially $\sum_{\boldsymbol{Z}\in\mathcal{Z}} h(\boldsymbol{Z}) \le \sum_{\delta\in\mathcal{D}} h(\delta)$ holds, with witness terms $\mathcal{M}$ and $\mathcal{S}$. After applying the change in Eq. (25), inequality $\sum_{\boldsymbol{Z}\in\mathcal{Z}} h(\boldsymbol{Z}) \le \sum_{\delta\in\overline{\mathcal{D}}} h(\delta)$ is still a Shannon inequality with the same witness. (Note that $\overline{\mathcal{D}}$ is defined in line 12). Now, we apply Lemma 3.8 to *drop* $\delta_0 = (\boldsymbol{YW}|\emptyset)$ from $\overline{\mathcal{D}}$, obtaining new parameters $(\mathcal{Z}', \mathcal{D}', \mathcal{M}', \mathcal{S}')$ to make progress on our algorithm. We remove from $\boldsymbol{T}$ and $\boldsymbol{n}$ the terms that were dropped from $\mathcal{D}$.

Case 2: Projection There exists a monotonicity term $(\boldsymbol{Y}|\boldsymbol{X}) \in \mathcal{M}$ with $\boldsymbol{W} = \boldsymbol{XY}$ (line 16). Here, the Shannon inequality is modified by applying the monotonicity-statistics cancellation in Eq. (26). This change is reflected in $\mathcal{M}'$ and $\mathcal{D}'$. The new statistics term $(\boldsymbol{X}|\emptyset) \in \mathcal{D}'$ has a guard which is the projection $T_{\boldsymbol{X}}$.

Case 3: Partition There exists a submodularity measure $(\boldsymbol{Y};\boldsymbol{Z}|\boldsymbol{X}) \in \mathcal{S}$ with $\boldsymbol{W} = \boldsymbol{XY}$ (line 22). In this case, instead of applying identity (27) from Lemma 3.8, we apply the following identity to maintain the Shannon inequality:

$$\begin{aligned} h(\boldsymbol{W}) - h(\boldsymbol{Y};\boldsymbol{Z}|\boldsymbol{X}) &= h(\boldsymbol{W}) - h(\boldsymbol{XY}) - h(\boldsymbol{XZ}) + h(\boldsymbol{X}) + h(\boldsymbol{XYZ}) \\ &= h(\boldsymbol{X}) + h(\boldsymbol{Y}|\boldsymbol{XZ}) \end{aligned} \tag{38}$$

Identity (38) says that we have two new statistics terms, $h(\boldsymbol{X}|\emptyset)$ and $h(\boldsymbol{Y}|\boldsymbol{XZ})$, which need guards; this is where we will need to create a logarithmic number of sub-problems in order to create the guards of the right statistics.

12 We shall show that an unconditional δ exists in Section 5.3.

Initialize the sub-problem tree with a single node with input parameters $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$
while $\exists$ a non-terminal leaf β **do** ▷ Tree-growing Loop
  Let $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$ be the parameters of this leaf
  Pick $\delta = (\boldsymbol{W}|\emptyset) \in \mathcal{D}$ arbitrarily ▷ See Prop 5.3 for why such δ exists
  if $\exists\ (\boldsymbol{Y}|\boldsymbol{W}) \in \mathcal{D}$ **then** ▷ Case 1 of Lemma 3.8: apply (25)
    if $n_{\boldsymbol{W}} + n_{\boldsymbol{Y}|\boldsymbol{W}} \le b_{\Delta,\boldsymbol{n}}$ **then**
      $\boldsymbol{n}' = \boldsymbol{n} \cup \{n_{\boldsymbol{YW}}\} - \{n_{\boldsymbol{W}} + n_{\boldsymbol{Y}|\boldsymbol{W}}\}$, where $n_{\boldsymbol{YW}} := n_{\boldsymbol{W}} + n_{\boldsymbol{Y}|\boldsymbol{W}}$
      $\mathcal{D}' = \mathcal{D} \cup \{(\boldsymbol{YW}|\emptyset)\} - \{(\boldsymbol{W}|\emptyset), (\boldsymbol{Y}|\boldsymbol{W})\}$
      $\boldsymbol{T}' = \boldsymbol{T} \cup \{T_{\boldsymbol{YW}}\} - \{T_{\boldsymbol{W}}, T_{\boldsymbol{Y}|\boldsymbol{W}}\}$, where $T_{\boldsymbol{YW}} := T_{\boldsymbol{W}} \bowtie T_{\boldsymbol{Y}|\boldsymbol{W}}$, which guards $(\boldsymbol{YW}|\emptyset)$
      Create a child of β, with parameters $(\mathcal{Z}, \mathcal{D}', \mathcal{M}, \mathcal{S}, \boldsymbol{T}', \boldsymbol{n}')$
    else
      $\overline{\mathcal{D}} = \mathcal{D} \cup \{(\boldsymbol{YW}|\emptyset)\} - \{(\boldsymbol{W}|\emptyset), (\boldsymbol{Y}|\boldsymbol{W})\}$
      Apply Lemma 3.8 to $(\mathcal{Z}, \overline{\mathcal{D}}, \mathcal{M}, \mathcal{S})$ to obtain $(\mathcal{Z}', \mathcal{D}', \mathcal{M}', \mathcal{S}')$ where $\mathcal{D}' \subseteq \overline{\mathcal{D}} - \{(\boldsymbol{YW}|\emptyset)\}$
      Let $\boldsymbol{T}', \boldsymbol{n}'$ be $\boldsymbol{T}, \boldsymbol{n}$ with only the terms corresponding to $\mathcal{D}'$ retained
      Create a child of β, with parameters $(\mathcal{Z}', \mathcal{D}', \mathcal{M}', \mathcal{S}', \boldsymbol{T}', \boldsymbol{n}')$
  else if $\exists (\boldsymbol{Y}|\boldsymbol{X}) \in \mathcal{M}$ with $\boldsymbol{W} = \boldsymbol{XY}$ **then** ▷ Case 2 of Lemma 3.8: apply (26)
    $\boldsymbol{n}' = \boldsymbol{n} \cup \{n_{\boldsymbol{X}}\} - \{n_{\boldsymbol{W}}\}$ where $n_{\boldsymbol{X}} := n_{\boldsymbol{W}}$
    $\mathcal{M}' = \mathcal{M} - \{(\boldsymbol{Y}|\boldsymbol{X})\}$
    $\mathcal{D}' = \mathcal{D} \cup \{(\boldsymbol{X}|\emptyset)\} - \{(\boldsymbol{W}|\emptyset)\}$
    $\boldsymbol{T}' = \boldsymbol{T} \cup \{T_{\boldsymbol{X}}\} - \{T_{\boldsymbol{W}}\}$ where $T_{\boldsymbol{X}} := \pi_{\boldsymbol{X}}(T_{\boldsymbol{W}})$, which guards $(\boldsymbol{X}|\emptyset)$
    Create a child of β, with parameters $(\mathcal{Z}, \mathcal{D}', \mathcal{M}', \mathcal{S}, \boldsymbol{T}', \boldsymbol{n}')$
  else if $\exists (\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X}) \in \mathcal{S}$ with $\boldsymbol{W} = \boldsymbol{XY}$ **then** ▷ Case 3 of Lemma 3.8: apply (38) instead of (27)
    Partition $T_{\boldsymbol{W}} = T^1 \cup T^2 \cup \cdots \cup T^k$ with $k := O(\log |T_{\boldsymbol{W}}|)$, using Lemma 5.1
    for $i \leftarrow 1$ to k **do**
      $\boldsymbol{n}^i = \boldsymbol{n} \cup \{n^i_{\boldsymbol{X}}, n^i_{\boldsymbol{Y}|\boldsymbol{XZ}}\} - \{n_{\boldsymbol{W}}\}$ where $n^i_{\boldsymbol{X}} := \log |\pi_{\boldsymbol{X}}(T^i)|$ and $n^i_{\boldsymbol{Y}|\boldsymbol{XZ}} := \log \deg_{T^i}(\boldsymbol{Y}|\boldsymbol{X})$
      $\mathcal{D}^i = \mathcal{D} \cup \{(\boldsymbol{X}|\emptyset), (\boldsymbol{Y}|\boldsymbol{XZ})\} - \{(\boldsymbol{W}|\emptyset)\}$
      $\mathcal{S}^i = \mathcal{S} - \{(\boldsymbol{Y}; \boldsymbol{Z}|\boldsymbol{X})\}$
      $\boldsymbol{T}^i = \boldsymbol{T} \cup \{T^i_{\boldsymbol{X}}, T^i_{\boldsymbol{Y}|\boldsymbol{XZ}}\} - \{T_{\boldsymbol{W}}\}$, where $T^i_{\boldsymbol{X}} := \pi_{\boldsymbol{X}}(T^i)$ and $T^i_{\boldsymbol{Y}|\boldsymbol{XZ}}$ is an extension of $T^i_{\boldsymbol{Y}|\boldsymbol{X}}$
      Create the ith-child of β, with parameters $(\mathcal{Z}, \mathcal{D}^i, \mathcal{M}, \mathcal{S}^i, \boldsymbol{T}^i, \boldsymbol{n}^i)$

$Q(\boldsymbol{Z}) \leftarrow \emptyset$ for all $Q(\boldsymbol{Z}) \in \Sigma_{\text{out}}$
for each terminal leaf β **do** ▷ Result-gathering phase
  Let $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$ be the parameters of this leaf
  Pick $\delta = (\boldsymbol{Z}|\emptyset) \in \mathcal{D}$ such that $\boldsymbol{Z} \in \Sigma_{\text{out}}$ ▷ By Definition 5.2
  $Q(\boldsymbol{Z}) \leftarrow Q(\boldsymbol{Z}) \cup T_{\boldsymbol{Z}}$

Algorithm 1. PANDA

In particular, the algorithm performs a "partitioning step": it *uniformizes* $T_{\boldsymbol{W}} (= T_{\boldsymbol{XY}})$ by applying Lemma 5.1. Each of the sub-problems has the corresponding statistics as defined in the for-loop starting at line 24. The table T^i is on variables $\boldsymbol{W} = \boldsymbol{XY}$. Thus, in order to have it guard the new term $(\boldsymbol{Y}|\boldsymbol{XZ})$, we will need to apply a dictionary extension to it (see Section 5.1). Note that (36) guarantees that $n^i_{\boldsymbol{X}} + n^i_{\boldsymbol{Y}|\boldsymbol{XZ}} \le n_{\boldsymbol{XY}} = n_{\boldsymbol{W}}$.

5.3 Proof of correctness and runtime analysis

In this section, we prove that the algorithm is correct and analyzes its runtime, to complete the proof of Theorem 4.1. Both of these tasks are accomplished by showing that PANDA maintains the invariants described below.

Recall that every sub-problem in the tree is parameterized by a tuple $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$. For any sub-problem β that is parameterized by $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$, define

$$J_\beta := \mathsf{Dom}^V \ltimes \bowtie_{\delta \in \mathcal{D}} T_\delta$$

to be the join of all its dictionary guards. At time t in the algorithm, let $\mathcal{L}_t$ denote the set of current leaf nodes in the sub-problem tree. While spawning new sub-problems, (we will show that) PANDA maintains the following invariants over all tuples $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$: (Recall that $N_\delta \stackrel{\text{def}}{=} 2^{n_\delta}$ for all $\delta \in \mathcal{D}$.)

$$\text{Shannon inequality:} \quad \text{Identity (23) holds w.r.t. the tuple } (\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}) \tag{39}$$

$$\text{Non-empty output:} \quad \mathcal{Z} \neq \emptyset \tag{40}$$

$$\text{Lossless join:} \quad \bowtie \Sigma_{\mathsf{in}} \subseteq \bigcup_{\beta \in \mathcal{L}_t} J_\beta \qquad \forall t \tag{41}$$

$$\text{Small tables:} \quad n_{\boldsymbol{Y}} \le b_{\Delta, \boldsymbol{n}}, \qquad \forall (\boldsymbol{Y}|\emptyset) \in \mathcal{D} \tag{42}$$

$$\text{Guarded statistics:} \quad T_\delta \models N_\delta, \qquad \forall \delta \in \mathcal{D} \tag{43}$$

$$\text{Upper bound:} \quad \frac{1}{|\mathcal{Z}|} \sum_{\delta \in \mathcal{D}} n_\delta \le b_{\Delta, \boldsymbol{n}} \tag{44}$$

We start by showing that, if the invariants hold, then the answer is correct with the desired runtime.

PROPOSITION 5.3. *Let $N \stackrel{def}{=} \max_{\delta \in \Delta} N_\delta$. Suppose the invariants above are satisfied, then PANDA returns a feasible solution to the input disjunctive datalog rule in time $O(B_{\Delta, \boldsymbol{N}} \cdot (\log N)^{|\mathcal{S}|})$, where $\mathcal{S}$ is the* input *multiset of submodularity measures.*

PROOF. For a given non-terminal leaf β, the number of unconditional statistics terms in $\mathcal{D}$ is at least $|\mathcal{Z}|$. This follows from Proposition 3.11 and invariant (39). Invariant (40) ensures that $|\mathcal{Z}| \ge 1$, and thus the unconditional statistics term δ exists for line 4 of the algorithm to proceed.

Thanks to invariant (39), at least one of the three cases considered in the main loop must hold. Invariant (43) guarantees that we can proceed to perform the join or the partition steps using corresponding tables / dictionaries when we need them in the algorithm. In summary, the body of the main loop can proceed as explained without getting stuck somewhere.

For every node in the sub-problem tree that the algorithm creates, with parameter tuple $(\mathcal{Z}, \mathcal{D}, \mathcal{M}, \mathcal{S}, \boldsymbol{T}, \boldsymbol{n})$, define the "potential" quantity $|\mathcal{D}| + |\mathcal{M}| + 2|\mathcal{S}|$ as in the proof of Lemma 3.8. Then, similar to what happens in the lemma's proof, the potential of the child is at least 1 less than the potential of the parent.[13][14] Thus, the depth of the sub-problem tree is bounded by the original potential $|\mathcal{D}| + |\mathcal{M}| + 2|\mathcal{S}|$. This proves that the algorithm terminates.

The time spent within each node β of the tree is dominated by either the join step $T_{\boldsymbol{YW}} := T_{\boldsymbol{W}} \bowtie T_{\boldsymbol{Y}|\boldsymbol{W}}$ in Case 1, the projection $T_{\boldsymbol{X}} := \pi_{\boldsymbol{X}}(T_{\boldsymbol{W}})$ in Case 2, or the partition step in Case 3 whose cost is $O(|T_{\boldsymbol{W}}|)$. In Case 2 and Case 3, the cost is bounded by $B_{\Delta,\boldsymbol{N}}$, thanks to invariant (42). In Case 1, the join is only computed if $n_{\boldsymbol{W}} + n_{\boldsymbol{Y}|\boldsymbol{W}} \le b_{\Delta,\boldsymbol{n}}$, thus it is also within the budget of $B_{\Delta,\boldsymbol{N}}$. Overall, the total time spent on all the nodes of the tree is bounded by $B_{\Delta,\boldsymbol{N}}$ times the number of nodes. As we observed above, the depth of the tree is at most $|\mathcal{D}| + |\mathcal{M}| + 2|\mathcal{S}|$. Every node has a fanout of either 1, or $k = O(\log B_{\Delta,\boldsymbol{N}}) = O(\log N)$. Every time the fanout is more than 1, the number of submodularity measures in $\mathcal{S}$ is reduced by 1. Thus, the total runtime in data complexity is $O(B_{\Delta,\boldsymbol{N}} \cdot (\log N)^{|\mathcal{S}|})$.

Last but not least, we prove that the answer computed starting at line 31 is correct, which means according to Definition 2.2 that for every tuple $\boldsymbol{t} \in \bowtie \Sigma_{\text{in}}$, there must exist $Q(\boldsymbol{Z}) \in \Sigma_{\text{out}}$ for which $\pi_{\boldsymbol{Z}}(\boldsymbol{t}) \in Q(\boldsymbol{Z})$. Note that Definition 2.2 does *not* require the converse to hold.[15] In particular, the output may contain tuples that are not in $\bowtie \Sigma_{\text{in}}$. In order to show that for every tuple $\boldsymbol{t} \in \bowtie \Sigma_{\text{in}}$, there exists $Q(\boldsymbol{Z}) \in \Sigma_{\text{out}}$ for which $\pi_{\boldsymbol{Z}}(\boldsymbol{t}) \in Q(\boldsymbol{Z})$, we rely on invariant (41). In particular, let $\mathcal{L}$ denote the set of all final leaf nodes in the sub-problem tree. Then, every $\boldsymbol{t} \in \bowtie \Sigma_{\text{in}}$ belongs to J_β for some $\beta \in \mathcal{L}$. Since β is a terminal leaf node, there must exist $\delta = (\boldsymbol{Z}|\emptyset) \in \mathcal{D}$ such that $Q(\boldsymbol{Z}) \in \Sigma_{\text{out}}$, and so $\pi_{\boldsymbol{Z}}(\boldsymbol{t}) \in Q(\boldsymbol{Z})$ thanks to line 35 of Algorithm 1. ■

PROPOSITION 5.4. *Algorithm 1 maintains invariants* (39) *to* (44) *throughout its execution.*

PROOF. We verify that every invariant from (39) to (44) holds one by one, by induction. For the input, only invariant (42) may not hold, because some input tables may be larger than the desired bound $B_{\Delta,\boldsymbol{N}}$. We deal with this situation by repeatedly applying the reset lemma as was done in the `else` branch of Case 1 (line 11), dropping input tables that are too large. After this pre-processing step to make sure that all invariants are satisfied initially, we verify that they remain satisfied by induction.

---

13 Unlike Case 3 of Lemma 3.8 where applying Eq. (27) reduces $|\mathcal{S}|$ by one and increases $|\mathcal{M}|$ by one, in Case 3 of the algorithm, we apply Eq. (38) which reduces $|\mathcal{S}|$ by one and increases $|\mathcal{D}|$ by one.

14 Note that the `else` branch of Case 1 (line 11) also reduces the potential by at least one, and applying the reset lemma in line 13 never increases the potential.

15 The converse will only become relevant in Section 6 when we discuss the evaluation of conjunctive queries.

Invariant (39) is guaranteed by constructing the multisets $(\mathcal{Z}', \mathcal{D}', \mathcal{M}', \mathcal{S}')$ for the subproblems while keeping (23) intact. This is easy to verify in all three cases as we apply Eq. (25), (26), (38) or Lemma 3.8.

For invariant 40, the only place where $\mathcal{Z}$ is changed is at line 15. This happens when $n_{\boldsymbol{W}} + n_{\boldsymbol{Y}|\boldsymbol{W}} > b_{\Delta,\boldsymbol{n}}$. Since inequality (44) holds (at the previous iteration), we have

$$b_{\Delta,\boldsymbol{n}} \geq \frac{1}{|\mathcal{Z}|} \sum_{\delta \in \mathcal{D}} n_\delta \geq \frac{1}{|\mathcal{Z}|}(n_{\boldsymbol{W}} + n_{\boldsymbol{Y}|\boldsymbol{W}}) > \frac{1}{|\mathcal{Z}|} \cdot b_{\Delta,\boldsymbol{n}}$$

It follows that $|\mathcal{Z}| \geq 2$. Thus, when applying Lemma 3.8, we end up with $|\mathcal{Z}'| \geq |\mathcal{Z}| - 1 \geq 1$, which means $\mathcal{Z}'$ is not empty.

Invariants (41), (42), (43), and (44) can be verified one by one by simple case analysis. Below, we highlight the most prominent cases:

— Case 1 of the algorithm maintains invariant (42) specifically because we only do the join $T_{\boldsymbol{YW}} := T_{\boldsymbol{W}} \bowtie T_{\boldsymbol{Y}|\boldsymbol{X}}$ when $n_{\boldsymbol{YW}} := n_{\boldsymbol{W}} + n_{\boldsymbol{Y}|\boldsymbol{W}} \leq b_{\Delta,\boldsymbol{n}}$.

— The `else` branch of Case 1 (line 11) maintains invariant (44) because of the following:

$$\begin{aligned}
\sum_{\delta \in \mathcal{D}'} n'_\delta = \sum_{\delta \in \mathcal{D}'} n_\delta &\leq \sum_{\delta \in \mathcal{D}} n_\delta - n_{\boldsymbol{W}} - n_{\boldsymbol{Y}|\boldsymbol{W}} && \text{(by Lemma 3.8)} \\
&< \sum_{\delta \in \mathcal{D}} n_\delta - b_{\Delta,\boldsymbol{n}} && (n_{\boldsymbol{W}} + n_{\boldsymbol{YW}} > b_{\Delta,\boldsymbol{n}}) \\
&\leq |\mathcal{Z}| \cdot b_{\Delta,\boldsymbol{n}} - b_{\Delta,\boldsymbol{n}} && \text{(by invariant (44) before)} \\
&\leq |\mathcal{Z}'| \cdot b_{\Delta,\boldsymbol{n}} && \text{(because } |\mathcal{Z}'| \geq |\mathcal{Z}| - 1)
\end{aligned}$$

— Case 3 of the algorithm maintains invariant (44) because Eq. (36) implies that $n^i_{\boldsymbol{X}} + n^i_{\boldsymbol{Y}|\boldsymbol{XZ}} \leq n_{\boldsymbol{XY}} = n_{\boldsymbol{W}}$. (Recall the definition of $n^i_{\boldsymbol{X}}$ and $n^i_{\boldsymbol{Y}|\boldsymbol{XZ}}$ in line 25.) ■

6. Answering Conjunctive Queries in Submodular-Width Time

The main aim of this section is to explain how PANDA can be used to compute a conjunctive query in time given by its submodular width (plus the output size). Recall the definition of a conjunctive query in Equation (3). In Marx's work [28], the submodular width was only defined for Boolean conjunctive queries (i.e., $\boldsymbol{F}$ is empty) where all input relations are set to be of size N. In Section 6.1, we generalize this notion to the case when $\boldsymbol{F}$ is arbitrary and the input relations satisfy given degree constraints, which are much more general than a single relation size, and also subsume functional dependencies. Marx [28] gave an algorithm that can answer Boolean conjunctive queries in time $O(f(|\boldsymbol{V}|) \cdot N^{c \cdot \mathsf{subw}(Q)})$ where $|\boldsymbol{V}|$ is the number of variables, f is some computable function, N is the input size, c is a constant greater than 1, and $\mathsf{subw}(Q)$ is the submodular width of Q. Therefore, Marx's algorithm establishes fixed-parameter tractability of the class of Boolean conjunctive queries where the submodular width

subw(Q) is bounded.[16] In contrast, in Section 6.2, we present our algorithm, which in addition to handling arbitrary $\boldsymbol{F}$ and degree constraints, removes the constant c from the runtime at the expense of introducing a polylogarithmic factor in N. In particular, our algorithm solves a Boolean conjunctive query in time $O(f_1(|\boldsymbol{V}|) \cdot N^{\mathsf{subw}(Q)} \cdot (\log N)^{f_2(|\boldsymbol{V}|)})$, where f_1 and f_2 are two computable functions. Due to the extra term $(\log N)^{f_2(|\boldsymbol{V}|)}$, our algorithm cannot be used to establish fixed-parameter tractability of the class of Boolean conjunctive queries with bounded submodular width. However, it is more useful than Marx's algorithm when considering the *fine-grained complexity* of these queries.

6.1 Width parameters for conjunctive queries under degree constraints

Given a conjunctive query Q in the form (3):

$$Q(\boldsymbol{F}) \text{ :- } \bigwedge_{R(\boldsymbol{X}) \in \Sigma} R(\boldsymbol{X}),$$

let $\mathcal{T}(Q)$ denote the set of all free-connex tree decompositions of the query.[17] (See Section 2.2 for the definition of a free-connex tree decomposition.) Let $(\Delta, \boldsymbol{N})$ be a set of degree constraints, as defined in Sec. 2.5. As usual, we denote by $(\Delta, \boldsymbol{n})$ the associated log-degree constraints. We say that a polymatroid $\boldsymbol{h}$ *satisfies* the constraints, and write $\boldsymbol{h} \models (\Delta, \boldsymbol{n})$, if $h(\delta) \leq n_\delta$ for all $\delta \in \Delta$.

DEFINITION 6.1. The *degree-aware fractional hypertree width* and the *degree-aware submodular width* of Q under the degree constraints $(\Delta, \boldsymbol{n})$ are:

$$\mathsf{fhtw}(Q, \Delta, \boldsymbol{n}) \stackrel{\text{def}}{=} \min_{(T,\chi)\in\mathcal{T}(Q)} \max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \max_{t\in\mathsf{nodes}(T)} h(\chi(t)) \tag{45}$$

$$\mathsf{subw}(Q, \Delta, \boldsymbol{n}) \stackrel{\text{def}}{=} \max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min_{(T,\chi)\in\mathcal{T}(Q)} \max_{t\in\mathsf{nodes}(T)} h(\chi(t)) \tag{46}$$

(Note that $\boldsymbol{h} \models (\Delta, \boldsymbol{n})$ requires $\boldsymbol{h}$ to both be a polymatroid and satisfy the degree constraints.)

REMARK 6.2. Eq. (45) and (46) collapse back to the standard definitions of the fractional hypertree width [22] and submodular width [28], respectively, when the degree constraints $(\Delta, \boldsymbol{N})$ only contain cardinality constraints of the form $|T_{\boldsymbol{Y}}| \leq N$ for a single number N that represents the input size; see [6] for a proof. Note, however, that in the standard definitions of fhtw and subw, the base of the log function was N, the input size, and thus runtimes were stated in the form $O(N^{\mathsf{fhtw}})$ and $O(N^{\mathsf{subw}})$. In our generalization, the base of the log function is 2, and thus runtimes are stated in the form $O(2^{\mathsf{fhtw}})$ and $O(2^{\mathsf{subw}})$.

---

16 Marx's work [**28**] also proves the converse but only for *self-join-free* queries, where a *self-join-free* query is a query whose body atoms have distinct relation symbols. In particular, Marx's work proves that the class of self-join-free Boolean conjunctive queries where the submodular width is unbounded is not fixed-parameter tractable, conditioned on the ETH conjecture.

17 For a fixed query Q, there are at most $2^{2^{|\boldsymbol{V}|}}$ tree decompositions, since any two trees that have the same set of bags can be considered equal for our purposes.

It is straightforward to use PANDA to answer a conjunctive query in time $\tilde{O}(\|\Sigma\|+2^{\mathsf{fhtw}})$, but we will only describe the algorithm for the submodular width. The key advantage of fhtw over subw is that we can answer sum-product queries over any semiring in time $\tilde{O}(\|\Sigma\|+2^{\mathsf{fhtw}})$ using variable elimination [4], since the query plan involves only *one* (optimal) tree decomposition.

We will also show how to use PANDA to support *constant-delay enumeration*[18] of the output of a conjunctive query after a preprocessing time of $\tilde{O}(\|\Sigma\| + 2^{\mathsf{subw}})$.[19] Another line of work [12] shows how to support constant-delay enumeration after a preprocessing time of $\tilde{O}(\|\Sigma\| + 2^{c\cdot\mathsf{subw}})$ for some constant $c > 1$.

The two definitions (45) and (46) differ only in the first two operations: min max v.s. max min. It is easy to see that subw $\le$ fhtw, as it follows immediately from the max-min inequality, which states that $\max_x \min_y f(x,y) \le \min_y \max_x f(x,y)$ for any function f. As mentioned above, the degree-aware submodular width generalizes the submodular width considering richer sets of statistics on the input data. The original definition assumed only cardinality constraints: there is one cardinality constraint for each input relation, and they are all equal. In that case, both subw and fhtw are finite. In our generalization, subw can be ∞, for example when $\Delta = \emptyset$. When no confusion arises, we will simply call fhtw and subw the fractional hypertree width and submodular width, dropping the term *degree-aware*.

EXAMPLE 6.3 (fhtw and subw of the k-cycle with $k \ge 4$). Consider the Boolean k-cycle query with $k \ge 4$:

$$Q() \text{ :- } R_{1,2}(X_1,X_2) \wedge \cdots \wedge R_{k-1,k}(X_{k-1},X_k) \wedge R_{k,1}(X_k,X_1).$$

Suppose we have input cardinality statistics $N := |R_{1,2}| = \cdots = |R_{k-1,k}| = |R_{k,1}|$. For instance, in the query where all input relations are the edge set of a graph, N is the number of edges. We will show that subw $\le (2 - 1/\lceil k/2\rceil)\log N$ and fhtw $\ge 2\log N$ for this query.

To show the bound on the fhtw, note that, in any tree decomposition (T,χ), there must be at least one bag $\chi(t)$ that contains some three consecutive variables $\{X_{i-1},X_i,X_{i+1}\}$ on the cycle. Fix i accordingly and consider the polymatroid $\boldsymbol{h}$ defined by:

$$h(\boldsymbol{X}) = |\boldsymbol{X} \cap \{X_{i-1},X_{i+1}\}| \cdot \log N \qquad \text{for all } \boldsymbol{X} \subseteq \{X_1,\ldots,X_k\}$$

It is straightforward to verify that this is a polymatroid with $h(\chi(t)) = 2\log N$ and $\boldsymbol{h} \models (\Delta,\boldsymbol{n})$.

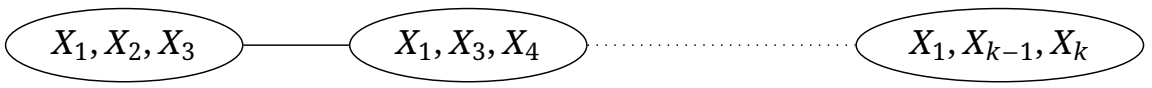
To prove the bound on subw, consider any polymatroid $\boldsymbol{h} \models (\Delta,\boldsymbol{n})$. Let θ be a parameter to be determined. Consider two cases.

---

18 By that, we mean reporting the output tuples one by one where the time needed to report the next tuple (or report that none exists) remains a constant throughout the entire process. The constant here is in *data complexity*.

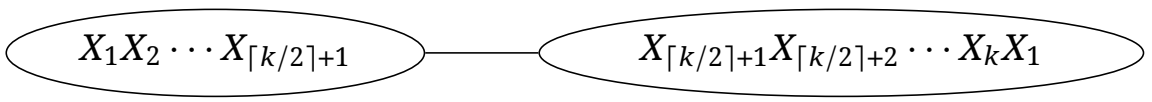
19 In particular, within the mentioned preprocessing time, we can produce a constant number of tree decompositions that cover the full join of the input relations. However, the same output tuple might be duplicated across multiple tree decompositions. Nevertheless, we can use the *Cheater's Lemma* [**15**] to deduplicate the output while maintaining constant delay. See the proof of Theorem 6.5 for more details.

— There exists X_i for which $h(X_i) \le \theta$. Without loss of generality, assume $h(X_1) \le \theta$. Consider the tree decomposition:



For every bag $B = \{X_1, X_i, X_{i+1}\}$, we have $h(B) \le h(X_1) + h(X_i X_{i+1}) \le \theta + \log N$.

— $h(X_i) > \theta$ for all $i \in [k]$. Consider the tree decomposition



From submodularity,

$$h(B_1) \le h(X_1X_2) + \sum_{i=3}^{\lceil k/2 \rceil + 1} h(X_i|X_{i-1}) \le \lceil k/2 \rceil \log N - (\lceil k/2 \rceil - 1)\theta$$

$$h(B_2) \le h(X_kX_1) + \sum_{i=\lceil k/2 \rceil + 1}^{k-1} h(X_i|X_{i+1}) \le \lfloor k/2 \rfloor \log N - (\lfloor k/2 \rfloor - 1)\theta.$$

Setting $\theta = (1 - 1/\lceil k/2 \rceil) \log N$ to balance the two cases, we conclude that subw $\le (2 - 1/\lceil k/2 \rceil) \log N$. ◆

6.2 Achieving submodular width runtime with PANDA

Before explaining how PANDA can be used to achieve the submodular width runtime, we need a technical lemma.

LEMMA 6.4. *Let $\mathcal{Z}$ denote a finite collection[20] of subsets of $\boldsymbol{V}$. Let $(\Delta, \boldsymbol{n})$ denote given input degree constraints. If the following quantity is finite:*

$$\mathsf{opt} := \max_{\boldsymbol{h} \models (\Delta, \boldsymbol{n})} \min_{\boldsymbol{Z} \in \mathcal{Z}} h(\boldsymbol{Z}), \tag{47}$$

then we can compute coefficients $\boldsymbol{\lambda} = (\lambda_{\boldsymbol{Z}})_{\boldsymbol{Z} \in \mathcal{Z}}$ and $\boldsymbol{w} = (w_\delta)_{\delta \in \Delta}$ such that the following are satisfied:

(a) $\|\boldsymbol{\lambda}\|_1 = 1$, $\boldsymbol{\lambda} \ge \boldsymbol{0}$, *and* $\boldsymbol{w} \ge \boldsymbol{0}$,

(b) Inequality $\sum_{\boldsymbol{Z} \in \mathcal{Z}} \lambda_{\boldsymbol{Z}} \cdot h(\boldsymbol{Z}) \le \sum_{\delta \in \Delta} w_\delta \cdot h(\delta)$ *is a Shannon inequality,*

(c) $\mathsf{opt} = \sum_{\delta \in \Delta} w_\delta n_\delta$.

PROOF. Let Γ denote the (polyhedral) set of all polymatroids over $\boldsymbol{V}$. We write opt in a slightly different form, where we introduce a new unconstrained variable t to replace the inner min:

$$\mathsf{opt} = \max_{t, \boldsymbol{h} \in \Gamma} \{t \mid \forall \boldsymbol{Z} \in \mathcal{Z} : t \le h(\boldsymbol{Z}), \text{ and } \forall \delta \in \Delta : h(\delta) \le n_\delta\} \tag{48}$$

20 Note that $\mathcal{Z}$ is *not* a multiset here.

Introduce a Lagrangian multiplier $\lambda_{\boldsymbol{Z}}$ for every constraint $t \le h(\boldsymbol{Z})$, and w_δ for every constraint $h(\delta) \le n_\delta$. The Lagrange dual function is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{w}) &= \max_{t, \boldsymbol{h} \in \Gamma} \left(t + \sum_{\boldsymbol{Z} \in \mathcal{Z}} \lambda_{\boldsymbol{Z}} (h(\boldsymbol{Z}) - t) + \sum_{\delta \in \Delta} w_\delta (n_\delta - h(\delta)) \right) \\ &= \sum_{\delta \in \Delta} w_\delta n_\delta + \max_t (1 - \|\boldsymbol{\lambda}\|_1) t + \max_{\boldsymbol{h} \in \Gamma} \left(\sum_{\boldsymbol{Z} \in \mathcal{Z}} \lambda_{\boldsymbol{Z}} h(\boldsymbol{Z}) - \sum_{\delta \in \Delta} w_\delta h(\delta) \right)\end{aligned}$$

Let $\boldsymbol{\lambda}^*$ and $\boldsymbol{w}^*$ denote an optimal solution to the Lagrangian dual problem $\min\{\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{w}) \mid \boldsymbol{\lambda} \ge \boldsymbol{0}, \boldsymbol{w} \ge \boldsymbol{0}\}$, then by strong duality[21]

$$\mathsf{opt} = \mathcal{L}(\boldsymbol{\lambda}^*, \boldsymbol{w}^*) = \sum_{\delta \in \Delta} w^*_\delta n_\delta + \max_t (1 - \|\boldsymbol{\lambda}^*\|_1) t + \max_{\boldsymbol{h} \in \Gamma} \left(\sum_{\boldsymbol{Z} \in \mathcal{Z}} \lambda^*_{\boldsymbol{Z}} h(\boldsymbol{Z}) - \sum_{\delta \in \Delta} w^*_\delta h(\delta) \right)$$

From the assumption that opt is finite, it follows that $\|\boldsymbol{\lambda}^*\|_1 = 1$ because t is unconstrained. Furthermore, if there is any polymatroid $\boldsymbol{h}$ for which $\sum_{\boldsymbol{Z} \in \mathcal{Z}} \lambda^*_{\boldsymbol{Z}} h(\boldsymbol{Z}) - \sum_{\delta \in \Delta} w^*_\delta h(\delta) > 0$ then $\mathcal{L}(\boldsymbol{\lambda}^*, \boldsymbol{w}^*)$ is unbounded, because any positive multiple of a polymatroid is a polymatroid. Thus, (b) is satisfied. Furthermore, as the expression inside $\max_{\boldsymbol{h} \in \Gamma}$ is non-positive, the maximum it can achieve is 0 with $\boldsymbol{h} = \boldsymbol{0}$. Consequently, $\boldsymbol{\lambda}^*$ and $\boldsymbol{w}^*$ satisfy the three conditions (a), (b), and (c) above, and we can compute them with standard linear programming algorithms. ■

Equipped with this tool, we are now ready to show how PANDA can be used to answer a conjunctive query in submodular width time:

THEOREM 6.5. *Given a set of degree constraints $(\Delta, \boldsymbol{N})$, a conjunctive query Q of the form (3) can be computed in time*

$$\tilde{O}(\|\Sigma\| + 2^{subw(Q, \Delta, \boldsymbol{n})} + |output|)$$

on any database instance Σ that satisfies the degree constraints. In particular, within a preprocessing time of $\tilde{O}(\|\Sigma\| + 2^{subw(Q, \Delta, \boldsymbol{n})})$, we can support constant-delay enumeration of the output.

PROOF. Let $\mathcal{T}(Q) = \{(T_1, \chi_1), \ldots, (T_m, \chi_m)\}$ be all free-connex tree decompositions of Q. For every tree decomposition $(T_i, \chi_i) \in \mathcal{T}(Q)$ and every node $j \in \mathsf{nodes}(T_i)$, create a fresh atom $A_{ij}(\boldsymbol{Z}_{ij})$ over variables $\boldsymbol{Z}_{ij} := \chi_i(j)$. In other words, every bag of every tree decomposition is associated with an atom. Let $\Sigma^i := \{A_{ij}(\boldsymbol{Z}_{ij}) \mid j \in \mathsf{nodes}(T_i)\}$ denote a schema corresponding to the bags of the ith tree decomposition. The algorithm will compute relation instances $A_{ij}(\boldsymbol{Z}_{ij})$, for all $i \in [m]$ and $j \in \mathsf{nodes}(T_i)$, such that the tree decompositions together cover the full join of the input relations:

$$\bowtie \Sigma \subseteq \bigcup_{i \in [m]} \bowtie \Sigma^i \tag{49}$$

21 Which holds because the problem is linear.

From these instances, there are m separate free-connex acyclic conjunctive queries of the form

$$Q^i(\boldsymbol{F}) \coloneq \bigwedge_{A_{ij}(\boldsymbol{Z}_{ij})\in\Sigma^i} A_{ij}(\boldsymbol{Z}_{ij}),$$

which can be computed in $\tilde{O}(\|\Sigma^i\| + |Q^i(\boldsymbol{F})|)$ time, using Lemma 2.1. Before computing these queries, we semijoin reduce each $A_{ij}(\boldsymbol{Z}_{ij})$ in Σ^i with all the input relations in Σ. Recall that by definition of a tree decomposition, every input relation $R(\boldsymbol{X})$ in Σ must have its variables $\boldsymbol{X}$ appear in some bag of the tree decomposition (T_i, χ_i), hence in some $\boldsymbol{Z}_{ij}$ in Σ^i. Therefore, this semijoin reduction ensures that the output of each query $Q^i(\boldsymbol{F})$ is a subset of the full join of the input relations $\bowtie\Sigma$. This turns (49) into an equality.

To support constant-delay enumeration of the output after a preprocessing time of $\tilde{O}(\|\Sigma\| + 2^{\mathsf{subw}(Q,\Delta,\boldsymbol{n})})$, we will use constant-delay enumeration of the output of each $Q^i(\boldsymbol{F})$ after a preprocessing time of $\tilde{O}(\|\Sigma^i\|)$, as explained in Lemma 2.1. Note that the same output tuple might be duplicated across multiple queries $Q^i(\boldsymbol{F})$. Nevertheless, we can use the *Cheater's Lemma* [15] to deduplicate the output while maintaining constant delay.[22]

It remains to show that we can compute all the instances Σ^i satisfying (49) in time $\tilde{O}(2^{\mathsf{subw}(Q,\Delta,\boldsymbol{n})})$. Obviously, to compute them in time $\tilde{O}(2^{\mathsf{subw}(Q,\Delta,\boldsymbol{n})})$, it is necessary that $\|\Sigma^i\| = \tilde{O}(2^{\mathsf{subw}(Q,\Delta,\boldsymbol{n})})$. To this end, for every combination of nodes $\boldsymbol{j} = (j_1, j_2, \ldots, j_m) \in \mathsf{nodes}(T_1) \times \cdots \times \mathsf{nodes}(T_m)$, we will compute a feasible output $B_1^{\boldsymbol{j}}, \ldots B_m^{\boldsymbol{j}}$ to the following DDR (whose input schema is Σ):

$$\bigvee_{i\in[m]} B_i^{\boldsymbol{j}}(\boldsymbol{Z}_{ij_i}) \coloneq \bigwedge_{R(\boldsymbol{X})\in\Sigma} R(\boldsymbol{X}). \qquad \text{(the } \boldsymbol{j}\text{th DDR)} \qquad (50)$$

In words, for this DDR, there is a representative bag $B_i^{\boldsymbol{j}}$ from each tree decomposition (T_i, χ_i). After feasible solutions to all these DDRs are computed, then we set $A_{ij} \coloneqq \bigcup_{\boldsymbol{j}:j_i=j} B_i^{\boldsymbol{j}}$.

We first prove that the instances A_{ij} defined as such satisfy property (49). Suppose there is a tuple $\boldsymbol{t} \in \bowtie\Sigma$ that is not in the RHS of (49). Then, for every $i \in [m]$, there exists a node $j_i \in \mathsf{nodes}(T_i)$ such that $\pi_{\boldsymbol{Z}_{ij_i}}(\boldsymbol{t}) \notin A_{ij_i}$. Collect these j_i into a tuple $\boldsymbol{j}$, then this implies that we did not compute a feasible output to the $\boldsymbol{j}$th DDR (50), a contradiction.

Last but not least, we show that the DDRs (50) can be computed in time $\tilde{O}(2^{\mathsf{subw}(Q,\Delta,\boldsymbol{n})})$. Fix a tuple of nodes $\boldsymbol{j} = (j_1, \ldots, j_m)$. Let $\mathcal{Z}$ denote the set of all bags $\boldsymbol{Z}_{ij_i}$ for $i \in [m]$. Define

$$\mathsf{opt} \coloneqq \max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min_{Z\in\mathcal{Z}} h(Z).$$

Then,

$$\mathsf{opt} = \max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min_{i\in[m]} h(Z_{ij_i}) \le \max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min_{i\in[m]} \max_{t\in\mathsf{nodes}(T_i)} h(\chi(t)) = \mathsf{subw}(Q,\Delta,\boldsymbol{n}).$$

22 Recall that the number of tree decompositions is constant in data complexity.

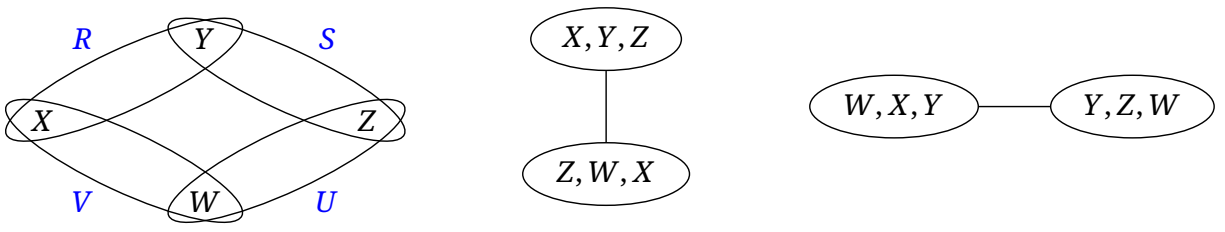


Figure 2. Query (51) with the two free-connex tree decompositions.

WLOG, we assume that subw is finite, which means opt is finite. By Lemma 6.4, we can compute coefficients $\boldsymbol{\lambda}$ and $\boldsymbol{w}$ such that the three conditions in Lemma 6.4 are satisfied. Hence, from Theorem 4.1, we can compute a feasible output to the DDR (50) in time $\tilde{O}(2^{\mathsf{opt}}) = \tilde{O}(2^{\mathsf{subw}(Q,\Delta,\boldsymbol{n})})$. ∎

6.3 Example: Solving a conjunctive query in submodular width time

Consider the following query Q whose body is a 4-cycle:

$$Q(X,Y) \coloneq R(X,Y) \wedge S(Y,Z) \wedge U(Z,W) \wedge V(W,X) \tag{51}$$

Suppose we only have cardinality statistics where all input relation sizes are upper bounded by N for some number N, i.e.,

$$\Delta = \{(XY|\emptyset), (YZ|\emptyset), (ZW|\emptyset), (WX|\emptyset)\}, \qquad N_{XY} = N_{YZ} = N_{ZW} = N_{WX} = N.$$

Let $n := \log N$. This query has two free-connex tree decompositions, depicted in Figure 2 (ignoring the trivial tree decomposition with a single bag):

- One with two bags $A_{11}(X,Y,Z)$ and $A_{12}(Z,W,X)$.
- One with two bags $A_{21}(Y,Z,W)$ and $A_{22}(W,X,Y)$.

It is not hard to see that the degree-aware fractional hypertree width of Q, given by (45), is exactly $2n$, and we leave this as an exercise. Next, we show that the degree-aware submodular width is $3n/2$. In particular, Eq. (46) for this query becomes:

$$\mathsf{subw}(Q,\Delta,\boldsymbol{n}) = \max_{\boldsymbol{h} \models (\Delta,\boldsymbol{n})} \min(\max(h(XYZ), h(ZWX)), \max(h(YZW), h(WXY)))$$

By distributing the min over the inner max, and then swapping the two max operators, we get:

$$\mathsf{subw}(Q,\Delta,\boldsymbol{n}) = \max($$
$$\max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min(h(XYZ), h(YZW)), \tag{52}$$
$$\max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min(h(XYZ), h(WXY)), \tag{53}$$
$$\max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min(h(ZWX), h(YZW)), \tag{54}$$
$$\max_{\boldsymbol{h}\models(\Delta,\boldsymbol{n})} \min(h(ZWX), h(WXY))) \tag{55}$$

Note that each of the expressions (52)…(55) has the same format as the optimization problem (47) in Lemma 6.4 and is equivalent to the linear program (48). Let's take the first expression (52). (The other three are similar.) For this expression, a linear program solver gives $\mathsf{opt} = 3n/2$. Lemma (6.4) guarantees for us the existence of the following Shannon inequality, which is the same as (32) from Section 4.2:

$$\frac{1}{2}h(XYZ) + \frac{1}{2}h(YZW) \le \frac{1}{2}h(XY|\emptyset) + \frac{1}{2}h(YZ|\emptyset) + \frac{1}{2}h(ZW|\emptyset)$$

In particular, by item (c) of the lemma, we have:

$$\mathsf{opt} = \frac{1}{2}n_{XY} + \frac{1}{2}n_{YZ} + \frac{1}{2}n_{ZW} = \frac{3}{2}n.$$

The other three expressions (53)–(55) also have $\mathsf{opt} = 3n/2$, leading to $\mathsf{subw}(Q,\Delta,\boldsymbol{n}) = 3n/2$.

Next, we describe the algorithm to compute the query (51) in time $\tilde{O}(N^{3/2} + |\text{output}|)$, as claimed in Theorem 6.5. The algorithm starts by constructing the following four DDRs, which mirror the four expressions (52)–(55):

$$B_1^{11}(X,Y,Z) \vee B_2^{11}(Y,Z,W) \text{ :- } R(X,Y) \wedge S(Y,Z) \wedge U(Z,W) \wedge V(W,X) \tag{56}$$
$$B_1^{12}(X,Y,Z) \vee B_2^{12}(W,X,Y) \text{ :- } R(X,Y) \wedge S(Y,Z) \wedge U(Z,W) \wedge V(W,X) \tag{57}$$
$$B_1^{21}(Z,W,X) \vee B_2^{21}(Y,Z,W) \text{ :- } R(X,Y) \wedge S(Y,Z) \wedge U(Z,W) \wedge V(W,X) \tag{58}$$
$$B_1^{22}(Z,W,X) \vee B_2^{22}(W,X,Y) \text{ :- } R(X,Y) \wedge S(Y,Z) \wedge U(Z,W) \wedge V(W,X) \tag{59}$$

Let's take the first DDR (56) as an example. We can compute a feasible output to this DDR by computing a feasible output to the following DDR instead:

$$B_1^{11}(X,Y,Z) \vee B_2^{11}(Y,Z,W) \text{ :- } R(X,Y) \wedge S(Y,Z) \wedge U(Z,W)$$

The above DDR is identical to (7), and we saw in Section 4.2 that we can compute a feasible output to it in time $\tilde{O}(N^{3/2})$. The other 3 DDRs (57)–(59) can be computed in the same way.

Afterwards, we compute:

$$A_{11} := B_1^{11} \cup B_1^{12}$$
$$A_{12} := B_1^{21} \cup B_1^{22}$$
$$A_{21} := B_2^{11} \cup B_2^{21}$$
$$A_{22} := B_2^{12} \cup B_2^{22}$$

Using Lemma (2.1), we compute the following two free-connex acyclic conjunctive queries (after *semijoin-reducing* each of the A_{ij} relations with the input relations R, S, U and V):

$$Q^1(X,Y) \text{ :- } A_{11}(X,Y,Z) \wedge A_{12}(Z,W,X)$$
$$Q^2(X,Y) \text{ :- } A_{21}(Y,Z,W) \wedge A_{22}(W,X,Y)$$

Finally, we take the union of Q^1 and Q^2 above as the output Q to the query in (51). The overall runtime is $\tilde{O}(N^{3/2} + |\text{output}|)$.

In order to prove the correctness of this algorithm, we show that the full join $R(X,Y) \bowtie S(Y,Z) \bowtie U(Z,W) \bowtie V(W,X)$ is identical to the set of tuples (x,y,z,w) that satisfy:

$$(A_{11}(x,y,z) \wedge A_{12}(z,w,x)) \vee (A_{21}(y,z,w) \wedge A_{22}(w,x,y)) \tag{60}$$

The containment $\supseteq$ is immediate from the definition of A_{ij} (and thanks to the semijoin reduction of A_{ij} with the input relations R, S, U and V). For the other containment $\subseteq$, note that condition (60) is equivalent to the following:

$$\begin{aligned}&(A_{11}(x,y,z) \vee A_{21}(y,z,w)) \wedge \\ &(A_{11}(x,y,z) \vee A_{22}(w,x,y)) \wedge \\ &(A_{12}(z,w,x) \vee A_{21}(y,z,w)) \wedge \\ &(A_{12}(z,w,x) \vee A_{22}(w,x,y))\end{aligned} \tag{61}$$

By (56)… (59), every tuple (x,y,z,w) that satisfies the conjunction $R(x,y) \wedge S(y,z) \wedge U(z,w) \wedge V(w,x)$ must also satisfy (61). This completes the proof of correctness.

7. Conclusion

We presented PANDA, an algorithm that computes a conjunctive query in time given by its submodular width. For this purpose, we have used a generalization of the notion of submodular width in [28], by incorporating a rich class of statistics on the input relations, including cardinality constraints and degree constraints; the latter can also express functional dependencies. The PANDA algorithm described here is a significant simplification of its preliminary version in [6]. PANDA can also be used as a Worst-Case-Optimal-Join algorithm to compute the output of a full conjunctive query in time bounded by the information-theoretic upper bound of the

output size. A recent extension showed that it can also be extended to account for ℓ_p-norms of degree sequences in the input [3].

We leave some open problems. The first is an analysis of the complexity of the witness of a Shannon inequality. The number of submodularities in the Shannon inequality appears as the exponent of a logarithmic factor in the runtime of PANDA, and it would be very useful to study this number as a function of the query. Another question concerns the number of tree decompositions needed to compute the submodular width: our current bound is double exponential, and the question is whether this can be reduced. Finally, one open problem is whether PANDA can be generalized to achieve information-theoretic bounds corresponding to *non-Shannon inequalities* [40, 39].

References

[1] **Serge Abiteboul, Richard Hull, and Victor Vianu**. Foundations of Databases. Addison-Wesley, 1995. URL (1)

[2] **Mahmoud Abo Khamis, Phokion G. Kolaitis, Hung Q. Ngo, and Dan Suciu**. Bag query containment and information theory. *ACM Trans. Database Syst.* 46(3):12:1–12:39, 2021. DOI (20)

[3] **Mahmoud Abo Khamis, Vasileios Nakos, Dan Olteanu, and Dan Suciu**. Join size bounds using lp-norms on degree sequences. *Proc. ACM Manag. Data*, 2(2), May 2024. DOI (3, 41)

[4] **Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra**. FAQ: questions asked frequently. *Proceedings of the 35th ACM Symposium on Principles of Database Systems, PODS 2016*, pages 13–28. ACM, 2016. DOI (1, 34)

[5] **Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu**. Computing join queries with functional dependencies. *Proceedings of the 35th ACM Symposium on Principles of Database Systems, PODS 2016*, pages 327–342. ACM, 2016. DOI (3, 5, 20)

[6] **Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu**. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? *Proceedings of the 36th ACM Symposium on Principles of Database Systems, PODS 2017*, pages 429–444. ACM, 2017. DOI ePrint (1, 3–5, 9, 14, 33, 40)

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

[8] **Noga Alon, Raphael Yuster, and Uri Zwick**. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. DOI (4)

[9] **Albert Atserias, Martin Grohe, and Dániel Marx**. Size bounds and query plans for relational joins. *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 739–748. IEEE Computer Society, 2008. DOI (2, 10)

[10] **Albert Atserias, Martin Grohe, and Dániel Marx**. Size bounds and query plans for relational joins. *SIAM J. Comput.* 42(4):1737–1767, 2013. DOI (2, 11)

[11] **Guillaume Bagan, Arnaud Durand, and Etienne Grandjean**. On acyclic conjunctive queries and constant delay enumeration. *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2007. DOI (7)

[12] **Christoph Berkholz and Nicole Schweikardt**. Constant Delay Enumeration with FPT-Preprocessing for Conjunctive Queries of Bounded Submodular Width. *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 58:1–58:15, 2019. DOI (34)

[13] **Béla Bollobás and Andrew Thomason**. Projections of bodies and hereditary properties of hypergraphs. *Bull. London Math. Soc.* 27(5):417–424, 1995. DOI (2)

[14] **Karl Bringmann and Egor Gorbachev**. A fine-grained classification of subquadratic patterns for subgraph listing and friends. *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC 2025. ACM, 2025. (23)

[15] **Nofar Carmeli and Markus Kröll**. On the enumeration complexity of unions of conjunctive queries. *ACM Trans. Database Syst.* 46(2), May 2021. DOI (34, 37)

[16] **F. R. K. Chung, R. L. Graham, P. Frankl, and J. B. Shearer**. Some intersection theorems for ordered sets and graphs. *J. Combin. Theory Ser. A*, 43(1):23–37, 1986. DOI (2, 10, 11, 14)

[17] **Thomas Eiter**, **Georg Gottlob**, **and Heikki Mannila**. Disjunctive datalog. *ACM Trans. Database Syst.* 22(3):364–418, 1997. DOI (4, 7)

[18] **Ehud Friedgut and Jeff Kahn**. On the number of copies of one hypergraph in another. *Israel J. Math.* 105:251–256, 1998. DOI (2, 4, 10)

[19] **Georg Gottlob**, **Gianluigi Greco**, **Nicola Leone**, **and Francesco Scarcello**. Hypertree decompositions: questions and answers. *Proceedings of the 35th ACM Symposium on Principles of Database Systems, PODS 2016*, pages 57–74. ACM, 2016. DOI (1, 3)

[20] **Georg Gottlob**, **Stephanie Tien Lee**, **and Gregory Valiant**. Size and treewidth bounds for conjunctive queries. *Proceedings of the Twenty-Eigth ACM Symposium on Principles of Database Systems, PODS 2009*, pages 45–54. ACM, 2009. DOI (3)

[21] **Georg Gottlob**, **Stephanie Tien Lee**, **Gregory Valiant**, **and Paul Valiant**. Size and treewidth bounds for conjunctive queries. *J. ACM*, 59(3):16:1–16:35, 2012. DOI (3)

[22] **Martin Grohe and Dániel Marx**. Constraint solving via fractional edge covers. *ACM Trans. Algorithms*, 11(1):4:1–4:20, 2014. DOI (1, 3, 33)

[23] **Martin Grohe and Dániel Marx**. Constraint solving via fractional edge covers. *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 289–298. ACM Press, 2006. URL (2, 10)

[24] **Alon Itai and Michael Rodeh**. Finding a minimum circuit in a graph. *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, STOC 1977*, pages 1–10. ACM, 1977. DOI (2)

[25] **Viktor Leis**, **Andrey Gubichev**, **Atanas Mirchev**, **Peter A. Boncz**, **Alfons Kemper**, **and Thomas Neumann**. How good are query optimizers, really? *Proc. VLDB Endow.* 9(3):204–215, 2015. DOI (2)

[26] **Jorge Lobo**, **Jack Minker**, **and Arcot Rajasekar**. Foundations of disjunctive logic programming. Logic Programming. MIT Press, 1992. (4)

[27] **L. H. Loomis and H. Whitney**. An inequality related to the isoperimetric inequality. *Bull. Amer. Math. Soc*, 55:961–962, 1949. (2)

[28] **Dániel Marx**. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):42:1–42:51, 2013. DOI (3, 32, 33, 40)

[29] **Jack Minker**. Overview of disjunctive logic programming. *Ann. Math. Artif. Intell.* 12(1-2):1–24, 1994. DOI (4)

[30] **Hung Q. Ngo**. On an information theoretic approach to cardinality estimation (invited talk). *25th International Conference on Database Theory, ICDT 2022*, volume 220 of *LIPIcs*, 1:1–1:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. DOI (2)

[31] **Hung Q. Ngo**. Worst-case optimal join algorithms: techniques, results, and open problems. *Proceedings of the 37th ACM Symposium on Principles of Database Systems*, PODS '18, pages 111–124. ACM, 2018. DOI (2, 20)

[32] **Hung Q. Ngo**, **Ely Porat**, **Christopher Ré**, **and Atri Rudra**. Worst-case optimal join algorithms. *J. ACM*, 65(3):16:1–16:40, 2018. DOI (2, 3, 23)

[33] **J. Radhakrishnan**. Entropy and counting. *Computational Mathematics, Modelling and Algorithms*, 2003. **J. C. Misra**, editor (10)

[34] **A. Schrijver**. Combinatorial Optimization - Polyhedra and Efficiency. Springer, 2003. (14, 15)

[35] **Dan Suciu**. Applications of information inequalities to database theory problems. *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–30, 2023. DOI (14, 20)

[36] **Todd L. Veldhuizen**. Triejoin: A simple, worst-case optimal join algorithm. *Proc. 17th International Conference on Database Theory (ICDT) 2014*, pages 96–106. OpenProceedings.org, 2014. DOI (2)

[37] **Mihalis Yannakakis**. Algorithms for acyclic database schemes. *Very Large Data Bases, 7th International Conference, VLDB 1981*, pages 82–94. IEEE Computer Society, 1981. (7, 20)

[38] **Raymond W. Yeung**. Information Theory and Network Coding. Springer Publishing Company, Incorporated, 1st edition, 2008. (8, 12)

[39] **Zhen Zhang and Raymond W Yeung**. On characterization of entropy function via information inequalities. *IEEE Transactions on Information Theory*, 44(4):1440–1452, 1998. (5, 41)

[40] **Zhen Zhang and Raymond W. Yeung**. A non-shannon-type conditional inequality of information quantities. *IEEE Trans. Information Theory*, 43(6):1982–1986, 1997. (5, 9, 41)