

# On Functional Aggregate Queries with Additive Inequalities

Mahmoud Abo Khamis  
relationalAI

Ryan R. Curtin  
relationalAI

Benjamin Moseley  
Carnegie Mellon University

Hung Q. Ngo  
relationalAI

XuanLong Nguyen  
University of Michigan

Dan Olteanu  
University of Oxford

Maximilian Schleich  
University of Oxford

## ABSTRACT

Motivated by fundamental applications in databases and relational machine learning, we formulate and study the problem of answering *functional aggregate queries* (FAQ) in which some of the input factors are defined by a collection of additive inequalities between variables. We refer to these queries as FAQ-AI for short.

To answer FAQ-AI in the Boolean semiring, we define *relaxed* tree decompositions and *relaxed* submodular and fractional hypertree width parameters. We show that an extension of the InsideOut algorithm using Chazelle’s geometric data structure for solving the semigroup range search problem can answer Boolean FAQ-AI in time given by these new width parameters. This new algorithm achieves lower complexity than known solutions for FAQ-AI. It also recovers some known results in database query answering.

Our second contribution is a relaxation of the set of polymatroids that gives rise to the *counting* version of the submodular width, denoted by  $\#subw$ . This new width is sandwiched between the submodular and the fractional hypertree widths. Any FAQ and FAQ-AI over one semiring can be answered in time proportional to  $\#subw$  and respectively to the relaxed version of  $\#subw$ .

We present three applications of our FAQ-AI framework to relational machine learning:  $k$ -means clustering, training linear support vector machines, and training models using

non-polynomial loss. These optimization problems can be solved over a database asymptotically faster than computing the join of the database relations.

## KEYWORDS

functional aggregate queries; inequality joins; in-database machine learning; non-polynomial loss

## ACM Reference Format:

Mahmoud Abo Khamis, Ryan R. Curtin, Benjamin Moseley, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2019. On Functional Aggregate Queries with Additive Inequalities. In *38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'19), June 30–July 5, 2019, Amsterdam, Netherlands*. ACM, New York, NY, USA, 18 pages. <https://doi.org/10.1145/3294052.3319694>

## 1 INTRODUCTION

We consider the problem of computing functional aggregate queries with inequality joins, or FAQ-AI queries for short. This is a fundamental computational problem that goes beyond databases: core computation for supervised and unsupervised machine learning can be formulated in FAQ-AI.

Inequalities occur naturally in scenarios involving temporal and spatial relationships between objects in databases. In a retail scenario (e.g., TPC-H), we would like to compute the revenue generated by a customer’s orders whose dates closely precede the ship dates of their lineitems. In streaming scenarios, we would like to detect patterns of events whose time stamps follow a particular order [13]. In spatial data management scenarios, we would like to retrieve objects whose coordinates are within a multi-dimensional range or in close proximity of other objects [25]. The evaluation of Core XPath queries over XML documents amounts to the evaluation of conjunctive queries with inequalities expressing tree relationships in the pre/post plane [16].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
PODS'19, June 30–July 5, 2019, Amsterdam, Netherlands  
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6227-6/19/06...\$15.00  
<https://doi.org/10.1145/3294052.3319694>

## 1.1 Motivating examples

A key insight of this paper is that the efficient computation of inequality joins can reduce the computational complexity of supervised and unsupervised machine learning.

*Example 1.1.* The  $k$ -means algorithm divides the input dataset  $G$  into  $k$  clusters of similar data points [19]. Each cluster  $G_i$  has a mean  $\mu_i \in \mathbb{R}^n$ , which is chosen according to the following optimization (similarity is defined here with respect to the  $\ell_2$  norm):

$$\min_{(G_1, \dots, G_k)} \sum_{i=1}^k \sum_{\mathbf{x} \in G_i} \|\mathbf{x} - \mu_i\|_2^2. \quad (1)$$

Let  $\mu_{i,\ell}$  be the  $\ell$ 'th component of mean vector  $\mu_i$ . For a data point  $\mathbf{x} \in G$ , the function  $c_{ij}$  computes the difference between the squares of the  $\ell_2$ -distances from  $\mathbf{x}$  to  $\mu_i$  and from  $\mathbf{x}$  to  $\mu_j$ :

$$c_{ij}(\mathbf{x}) = \sum_{\ell \in [n]} [\mu_{i,\ell}^2 - 2x_\ell(\mu_{i,\ell} - \mu_{j,\ell}) - \mu_{j,\ell}^2].$$

A data point  $\mathbf{x} \in G$  is closest to mean  $\mu_i$  from the set of  $k$  means iff  $\forall j \in [k] : c_{ij}(\mathbf{x}) \leq 0$ .

To compute the mean vector  $\mu_i$ , we need to compute the sum of values for each dimension  $\ell \in [n]$  over  $G_i : \sum_{\mathbf{x} \in G_i} x_\ell$ . If the dataset  $G$  is the join of database relations  $(R_p)_{p \in [m]}$  over schemas  $S_p \subseteq [n]$ ,  $\forall p \in [m]$ , we can formulate this sum computation as a datalog-like query with aggregates [17]:

$$Q_1^{(i,\ell)} \left( \sum x_\ell \right) \leftarrow \left( \bigwedge_{p \in [m]} R_p(x_{S_p}) \right) \wedge \left( \bigwedge_{j \in [k]} c_{ij}(\mathbf{x}) \leq 0 \right).$$

The above notation means that the answer to query  $Q_1^{(i,\ell)}$  is the sum of  $x_\ell$  over all tuples  $\mathbf{x}$  satisfying the conjunction on the right-hand side. Section 4 gives further queries necessary to compute the  $k$ -means. As we show in this paper, such queries with aggregates and inequalities can be computed asymptotically faster than the join defining  $G$ .  $\square$

Simple queries can already highlight the limitations of current evaluation techniques, as shown next.

*Example 1.2.* State-of-the-art techniques take time  $O(N^2)$  to compute the following query over relations of size  $\leq N$ :

$$Q_2() \leftarrow R(a, b) \wedge S(b, c) \wedge T(c, d) \wedge a \leq d,$$

Examples 3.9 and 3.19 show how to compute  $Q_2$  and its counting version in time  $O(N^{1.5} \log N)$  using the techniques introduced in this paper.  $\square$

## 1.2 The FAQ-AI problem

One way to answer the above queries is to view them as *functional aggregate queries* (FAQ) [5] formulated in sum-product form over some semiring. We therefore briefly introduce FAQ over a single semiring.

First we establish notation. For any positive integer  $n$ , let  $\mathcal{V} = [n]$ . For  $i \in \mathcal{V}$ , let  $X_i$  denote a variable/attribute, and  $x_i$  denote a value in the discrete domain  $\text{Dom}(X_i)$  of  $X_i$ . For any  $K \subseteq \mathcal{V}$ , define  $\mathbf{X}_K = (X_i)_{i \in K}$ ,  $\mathbf{x}_K = (x_i)_{i \in K} \in \prod_{i \in K} \text{Dom}(X_i)$ . That is,  $\mathbf{X}_K$  is a tuple of variables and  $\mathbf{x}_K$  is a tuple of values for these variables.

Let a semiring  $(\mathcal{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$  and a **multi-hypergraph**  $\mathcal{H} = (\mathcal{V} = [n], \mathcal{E})$ . To each edge  $K \in \mathcal{E}$  we associate a function  $R_K : \prod_{v \in K} \text{Dom}(X_i) \rightarrow \mathcal{D}$  called *factor*. An FAQ query over one semiring with free variables  $F \subseteq \mathcal{V}$  has the form:

$$Q(\mathbf{x}_F) = \bigoplus_{\mathbf{x}_{\mathcal{V} \setminus F} \in \prod_{i \in \mathcal{V} \setminus F} \text{Dom}(X_i)} \bigotimes_{K \in \mathcal{E}} R_K(\mathbf{x}_K). \quad (2)$$

Under the Boolean semiring  $(\{\text{true}, \text{false}\}, \vee, \wedge, \text{false}, \text{true})$ , the query (2) becomes a conjunctive query: The factors  $R_K$  represent input relations, where  $R_K(\mathbf{x}_K) = \text{true}$  iff  $\mathbf{x}_K \in R_K$ , with some notational overloading. Under the sum-product semiring, the query (2) counts the number of tuples in the join result for each tuple  $\mathbf{x}_F$ , where the factors  $R_k$  are indicator functions  $R_K(\mathbf{x}_K) = \mathbf{1}_{\mathbf{x}_K \in R_K}$ . To aggregate over some input variable, say  $X_k$ , we can designate an identity factor  $R_k(x_k) = x_k$ .

Throughout the paper, we assume the query size to be a constant and state runtimes in data complexity. It is known [5] that over an arbitrary semiring, the query (2) can be answered in time  $O(N^{\text{fhtw}(Q)} \cdot \log N)$ , where *fhtw* denotes the *fractional hypertree width* of the query and  $Q$  has no free variables [15]. If  $Q$  has free variables, *fhtw*-width becomes *FAQ-width* instead [5]. Here  $N$  is the size of the largest factor  $R_K$ . Over the Boolean semiring, the time can be lowered to  $\tilde{O}(N^{\text{subw}(Q)})$  [7], where *subw* is the *submodular width* [26] and  $\tilde{O}$  hides a polylogarithmic factor in  $N$ .

Motivated by the examples in Section 1.1, we formulate a class of FAQ queries called **FAQ-AI**: the hyperedge multiset  $\mathcal{E}$  is partitioned into two multisets  $\mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_\ell$ , where  $s$  stands for “skeleton” and  $\ell$  stands for “ligament”. The input to our class of queries consists of the following: (1) to each hyperedge  $K \in \mathcal{E}_s$ , there corresponds a function  $R_K : \prod_{i \in K} \text{Dom}(X_i) \rightarrow \mathcal{D}$ , as in the FAQ case; (2) to each hyperedge  $S \in \mathcal{E}_\ell$ , there corresponds  $|S|$  functions  $\theta_v^S : \text{Dom}(X_v) \rightarrow \mathbb{R}$ , one for every variable  $v \in S$ . The query we want to compute is the following:

$$Q(\mathbf{x}_F) = \bigoplus_{\mathbf{x}_{\mathcal{V} \setminus F}} \left( \bigotimes_{K \in \mathcal{E}_s} R_K(\mathbf{x}_K) \right) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right). \quad (3)$$

The summation  $\bigoplus$  is over tuples  $\mathbf{x}_{\mathcal{V} \setminus F} \in \prod_{i \in \mathcal{V} \setminus F} \text{Dom}(X_i)$ . The notation  $\mathbf{1}_A$  denotes the indicator function of the event  $A$  in the semiring  $(\mathcal{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ :  $\mathbf{1}_A = \mathbf{1}$  if  $A$  holds, and  $\mathbf{0}$  otherwise. The (uni-variate) functions  $\theta_v^S$  can be user-defined functions, e.g.,  $\theta_1^S(x_1) = x_1^2/2$ , or binary predicates with one key in  $\text{Dom}(X_v)$  and a numeric value, e.g., a table

salary(employee\_id, salary\_value) where employee\_id is a key. The only requirement we impose is that, given  $x$ , the value  $\theta_v^S(x)$  can be accessed/computed in  $O(1)$ -time.

If  $\mathcal{E}_\ell = \emptyset$ , then we get back the FAQ formulation (2).

*Example 1.3.* The queries in Section 1.1 are instances of (3):

$$Q_1^{(i,\ell)}() = \bigoplus_{x_{[n]}} x_\ell \otimes \left( \bigotimes_{p \in [m]} R_p(\mathbf{x}_{S_p}) \right) \otimes \left( \bigotimes_{j \in [k]} \mathbf{1}_{c_{ij}(\mathbf{x}) \leq 0} \right), \quad (4)$$

$$Q_2() = \bigoplus_{x_{[4]}} R(x_1, x_2) \otimes S(x_2, x_3) \otimes T(x_3, x_4) \otimes \mathbf{1}_{x_1 - x_4 \leq 0}.$$

$Q_1$  is on the sum-product semiring.  $Q_2$  can be on any semiring: Example 3.9 discusses the case of the Boolean semiring while Example 3.19 discusses the sum-product semiring.  $\square$

### 1.3 Our contributions

To answer FAQ queries of the form (2), currently there are two dominant width parameters: fractional hypertree width (fhtw [15]) and submodular width (subw [26]).<sup>1</sup> It is known that  $\text{subw} \leq \text{fhtw}$  for any query, and in the Boolean semiring we can answer (2) in  $\tilde{O}(N^{\text{subw}})$ -time [7, 26]. For non-Boolean semirings, the best known algorithm, called InsideOut [5, 6], evaluates (2) in time  $O(N^{\text{fhtw}} \log N)$ . For queries with free variables, fhtw is replaced by the more general notion of FAQ-width (faqw) [5]; however, for brevity we discuss the non-free variable case here.

Following [6], both width parameters subw and fhtw can be defined via two constraint sets: the first is the set TD of all tree decompositions of the query hypergraph  $\mathcal{H}$ , and the second is the set of polymatroids  $\Gamma_n$  on  $n$  vertices of  $\mathcal{H}$ . The widths subw and fhtw are then defined as maximin or minimax optimization problems on the domain pair TD and  $\Gamma_n$ , subject to “edge domination” constraints for  $\Gamma_n$ . Section 2 presents these notions and other related preliminary concepts in detail.

Our contributions include the following:

*Answering FAQ-AI over Boolean semiring.* On the Boolean semiring, one way to answer query (3) is to apply the PANDA algorithm [26], using edge domination constraints on  $\mathcal{E}_s$  and the set TD of all tree decompositions of  $\mathcal{H} = (\mathcal{V}, \mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_\ell)$ . However, we can do better. In Section 3.2 we define a new notion of tree decomposition: *relaxed tree decomposition*, in which the hyperedges in  $\mathcal{E}_\ell$  only have to be covered by adjacent TD bags. Then, we present a variant of the InsideOut algorithm running on these relaxed TDs, using Chazelle’s classic geometric data structure [10] for solving the semigroup range search problem. We show that our InsideOut variant meets the “relaxed fhtw” runtime,

which is the analog of fhtw on relaxed TD. The PANDA algorithm can use the InsideOut variant as a blackbox to meet the “relaxed subw” runtime. The relaxed widths are smaller than the non-relaxed counterparts, and are strictly smaller for some classes of queries, which means our algorithms yield asymptotic improvements over existing ones.

*Answering FAQ over an arbitrary semiring.* Next, to prepare the stage for answering FAQ-AI over an arbitrary semiring, in Section 3.3 we revisit FAQ over a non-Boolean semiring, where no known algorithm can achieve the subw-runtime. Here, we relax the set  $\Gamma_n$  of polymatroids to a superset  $\Gamma'_n$  of *relaxed polymatroids*. Then, by adapting the subw definition to relaxed polymatroids, we obtain a new width parameter called “sharp submodular width” (#subw). We show how a variant of PANDA, called #PANDA, can achieve a runtime of  $\tilde{O}(N^{\#\text{subw}})$  for evaluating FAQ over an arbitrary semiring. We prove that  $\text{subw} \leq \#\text{subw} \leq \text{fhtw}$ , and that there are classes of queries for which #subw is unboundedly smaller than fhtw.

*Answering FAQ-AI over an arbitrary semiring.* Getting back to FAQ-AI, we apply the #subw result under both relaxations: relaxed TD and relaxed polymatroids, to obtain a new width parameter called the relaxed #subw. We show that the new variants of PANDA and InsideOut can achieve the relaxed #subw runtime. We also show that there are queries for which relaxed #subw is essentially the best we can hope for, modulo  $k$ -sum-hardness.

*Applications in relational Machine Learning.* Equipped with the algorithms for answering FAQ-AI, in Section 4 we return to relational machine learning applications over datasets defined by feature extraction queries over relational databases. We show how one can train linear SVM,  $k$ -means, and ML models using Huber/hinge loss functions without completely materializing the output of the feature extraction queries. In particular, this shows that for these important classes of ML models, one can sometimes train models in time sub-linear in the training dataset size.

### 1.4 Related work

The full version of the paper [1] revisits two prior results on the evaluation of queries with inequalities through FAQ-AI lenses: Core XPath queries over XML documents and inequality joins over tuple-independent probabilistic databases [30]. Throughout the paper, we contrast our new width notions with fhtw and subw and our new algorithm #PANDA with the state-of-the-art algorithms PANDA and InsideOut for FAQ and FAQ-AI queries.

A seminal work considers the containment and minimization problem for queries with inequalities [22]. Efficient evaluation of such queries continues to receive good attention

<sup>1</sup>Section 2.1 overviews other notions of widths.

in the database community [21]. There is a bulk of work on queries with *disequalities* (not-equal), e.g., [4, 23], which are at times referred to as *inequalities*. Queries with disequalities are a proper subclass of FAQ-AI (since  $x \neq y$  can be represented as  $x < y \vee x > y$ ). Prior works [4, 23] present several results for this proper subclass that are stronger than our general results for FAQ-AI in this work. In particular, for queries with disequalities it suffices to consider tree decompositions only for “skeleton” edges (ignoring “ligament” edges which -in this case- are the disequalities) [4, 23], whereas for the more general FAQ-AI we need to consider “relaxed” tree decompositions (see Def. 3.3).

Section 4 reviews relevant works on machine learning.

## 2 PRELIMINARIES

We assume without loss of generality in the paper that semiring operations  $\oplus$  and  $\otimes$  can be performed in  $O(1)$ -time. (When the assumption does not hold, for the set semiring for instance, we can multiply the claimed runtime with the real operation’s runtime.) All missing proofs and details can be found in the full version of the paper [1].

### 2.1 Tree decompositions and polymatroids

We briefly define tree decompositions, fhtw and subw parameters. We refer the reader to the recent survey by Gottlob et al. [14] for more details and historical contexts. In what follows, the hypergraph  $\mathcal{H}$  should be thought of as the hypergraph of the input query, although the notions of tree decomposition and width parameters are defined independently of queries.

A *tree decomposition* of a hypergraph  $\mathcal{H} = (\mathcal{V}, \mathcal{E})$  is a pair  $(T, \chi)$ , where  $T$  is a tree and  $\chi : V(T) \rightarrow 2^{\mathcal{V}}$  maps each node  $t$  of the tree to a subset  $\chi(t)$  of vertices such that

- (1) every hyperedge  $S \in \mathcal{E}$  is a subset of some  $\chi(t)$ ,  $t \in V(T)$  (i.e. every edge is covered by some bag),
- (2) for every vertex  $v \in \mathcal{V}$ , the set  $\{t \mid v \in \chi(t)\}$  is a non-empty (connected) sub-tree of  $T$ . This is called the *running intersection property*.

The sets  $\chi(t)$  are called the *bags* of the tree decomposition.

Let  $\text{TD}(\mathcal{H})$  denote the set of all tree decompositions of  $\mathcal{H}$ . When  $\mathcal{H}$  is clear from context, we use TD for brevity.

To define width parameters, we use the polymatroid characterization from Abo Khamis et al. [7]. A function  $f : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$  is called a (non-negative) *set function* on  $\mathcal{V}$ . A set function  $f$  on  $\mathcal{V}$  is *modular* if  $f(S) = \sum_{v \in S} f(\{v\})$  for all  $S \subseteq \mathcal{V}$ , is *monotone* if  $f(X) \leq f(Y)$  whenever  $X \subseteq Y$ , and is *submodular* if  $f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y)$  for all  $X, Y \subseteq \mathcal{V}$ . A monotone, submodular set function  $h : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$  with  $h(\emptyset) = 0$  is called a *polymatroid*. Let  $\Gamma_n$  denote the set of all polymatroids on  $\mathcal{V} = [n]$ .

Given  $\mathcal{H}$ , define the set of *edge dominated* set functions:

$$\text{ED} := \{h \mid h : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+, h(S) \leq 1, \forall S \in \mathcal{E}\}. \quad (5)$$

With this, we define the submodular width and fractional hypertree width of a given hypergraph  $\mathcal{H}$ :

$$\text{fhtw}(\mathcal{H}) := \min_{(T, \chi) \in \text{TD}} \max_{h \in \text{ED} \cap \Gamma_n} \max_{t \in V(T)} h(\chi(t)), \quad (6)$$

$$\text{subw}(\mathcal{H}) := \max_{h \in \text{ED} \cap \Gamma_n} \min_{(T, \chi) \in \text{TD}} \max_{t \in V(T)} h(\chi(t)). \quad (7)$$

It is known [26] that  $\text{subw}(\mathcal{H}) \leq \text{fhtw}(\mathcal{H})$ , and there are classes of hypergraphs with bounded subw and unbounded fhtw. Furthermore, fhtw is strictly less than other width notions such as (generalized) hypertree width and tree width.

**Remark 2.1.** Prior to Abo Khamis et al. [7], the commonly used definition of  $\text{fhtw}(\mathcal{H})$  is

$$\text{fhtw}(\mathcal{H}) := \min_{(T, \chi) \in \text{TD}} \max_{t \in V(T)} \rho_{\mathcal{E}}^*(\chi(t)),$$

where  $\rho_{\mathcal{E}}^*(B)$  is the fractional edge cover number of a vertex set  $B$  using the hyperedge set  $\mathcal{E}$ . It is straightforward to show, using linear programming duality [7], that

$$\max_{t \in V(T)} \max_{h \in \text{ED} \cap \Gamma_n} h(\chi(t)) = \max_{t \in V(T)} \rho_{\mathcal{E}}^*(\chi(t)), \quad (8)$$

proving the equivalence of the two definitions. However, the characterization (6) has two primary advantages: (i) it exposes the minimax / maximin duality between fhtw and subw, and more importantly (ii) it makes it completely straightforward to relax the definitions by replacing the  $\text{ED} \cap \Gamma_n$  constraints by other applicable constraints, as shall be shown in later sections.  $\square$

**Definition 2.2** (*F*-connex tree decomposition [8, 31]). Given a hypergraph  $\mathcal{H} = (\mathcal{V}, \mathcal{E})$  and a set  $F \subseteq \mathcal{V}$ , a tree decomposition  $(T, \chi)$  of  $\mathcal{H}$  is *F*-connex if  $F = \emptyset$  or the following holds: There is a nonempty subset  $V' \subseteq V(T)$  that forms a connected subtree of  $T$  and satisfies  $\bigcup_{t \in V'} \chi(t) = F$ .

We use  $\text{TD}_F$  to denote the set of all *F*-connex tree decompositions of  $\mathcal{H}$ . (Note that when  $F = \emptyset$ ,  $\text{TD}_F = \text{TD}$ .)

### 2.2 InsideOut and PANDA

To answer the FAQ query (2), we need a model for the representation of the input factors  $R_K$ . The support of the function  $R_K$  is the set of tuples  $\mathbf{x}_K$  such that  $R(\mathbf{x}_K) \neq \mathbf{0}$ . We use  $|R_K|$  to denote the size of its support. For example, if  $R_K$  represents an input relation, then  $|R_K|$  is the number of tuples in  $R_K$ . In practice, there often are factors with infinite support, e.g.,  $R_K$  represents a built-in function in a database, an arithmetic operator, or a comparison operator as in (3). To deal with this more general setting, the edge set  $\mathcal{E}$  is partitioned into two sets  $\mathcal{E} = \mathcal{E}_{\neq \infty} \cup \mathcal{E}_{\infty}$ , where  $|R_K|$  is finite for all  $K \in \mathcal{E}_{\neq \infty}$  and  $|R_K| = \infty$  for all  $K \in \mathcal{E}_{\infty}$ . For simplicity, we often state runtimes of algorithms in terms of the “input

size”  $N := \max_{K \in \mathcal{E}_{\phi_0}} |R_K|$ . Moreover, we use  $|Q|$  to denote the output size of  $Q$ . We always assume that  $\bigcup_{S \in \mathcal{E}_{\phi_0}} S = \mathcal{V}$ ; otherwise the output size  $|Q|$  could be infinite.

InsideOut [5, 6]. To answer (2), the InsideOut algorithm works by eliminating variables, along with an idea called the “indicator projection”. Its runtime is described by the FAQ-width of the query, a slight generalization of fhtw. In the context of one semiring, we can define  $\text{faqw}(Q)$  by applying Definition (6) over a restricted set of tree decompositions and edge dominated polymatroids. In particular, let  $F \subseteq \mathcal{V}$  denote the set of free variables in (2), and recall  $\text{TD}_F$  from Definition 2.2. Then,

$$\text{ED}_{\phi_0} := \{h \mid h : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+, h(S) \leq 1, \forall S \in \mathcal{E}_{\phi_0}\}, \quad (9)$$

$$\text{faqw}(Q) := \min_{(T, \chi) \in \text{TD}_F} \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n} \max_{t \in V(T)} h(\chi(t)) \quad (10)$$

$$\text{(remark 2.1)} = \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} \rho_{\mathcal{E}_{\phi_0}}^*(\chi(t)) \quad (11)$$

Note that  $\text{faqw}(Q) = \text{fhtw}(\mathcal{H})$  when  $F = \emptyset$  and  $\mathcal{E}_{\infty} = \emptyset$  (i.e.  $\mathcal{E} = \mathcal{E}_{\phi_0}$ ). A simple result from Abo Khamis et al. [5] is the following: (Recall that throughout the paper we assume the query size to be a constant and state runtimes in data complexity.)

**Proposition 2.3.** *InsideOut answers query (2) in time  $O(N^{\text{faqw}(Q)} \cdot \log N + |Q|)$ .*

To solve the FAQ-AI (3), we can apply Proposition 2.3 with  $\mathcal{E}_{\infty} \supseteq \mathcal{E}_{\ell}$  (because all ligament factors are infinite). But this is suboptimal—later, we show a new InsideOut variant that is polynomially better.

PANDA [7]. In case of the Boolean semiring, i.e., when the FAQ query (2) is of the form

$$Q(\mathbf{x}_F) = \bigvee_{\mathbf{x}_{\mathcal{V}_1 F} \in \prod_{i \in \mathcal{V}_1 F} \text{Dom}(X_i)} \bigwedge_{K \in \mathcal{E}} R_K(\mathbf{x}_K), \quad (12)$$

we can do much better than Proposition 2.3. When  $F = \emptyset$ , Marx [26] showed that (12) can be answered in time  $\tilde{O}(N^{O(\text{subw}(Q))})$ . The PANDA algorithm [7] generalizes Marx’s result to deal with general degree constraints, and to meet precisely the  $\tilde{O}(N^{\text{subw}(Q)})$ -runtime. In fact, PANDA works with queries such as (12) with free variables as well. In the context of this paper, we can define the following notion of *submodular FAQ-width* in a very natural way:

$$\text{smfw}(Q) := \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n} \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} h(\chi(t)). \quad (13)$$

Then, the results from Abo Khamis et al. [7] imply:

**Proposition 2.4.** *PANDA answers query (12) in time  $\tilde{O}(N^{\text{smfw}(Q)} + |Q|)$ .*

These results only work for the Boolean semiring. Section 3 introduces a variant of PANDA, called #PANDA, that also works for non-Boolean semirings.

## 2.3 Semigroup range searching

Orthogonal range counting (and searching) is a classic and ubiquitous problem in computational geometry [12]: given a set  $S$  of  $N$  points in a  $d$ -dimensional space, build a data structure that, given any  $d$ -dimensional rectangle, can efficiently return the number of enclosed points. More generally, there is the semigroup range searching problem [10], where each point  $\mathbf{p} \in S$  of the  $N$  input points also has a weight  $w(\mathbf{p}) \in G$ , where  $(G, \oplus)$  is a *semigroup*.<sup>2</sup> The problem is: given a  $d$ -dimensional rectangle  $R$ , compute  $\bigoplus_{\mathbf{p} \in S \cap R} w(\mathbf{p})$ .

Classic results by Chazelle [10] show that there are data structures for semigroup range searching which can be constructed in time  $O(N \log^{d-1} N)$ , and answer rectangular queries in  $O(\log^{d-1} N)$ -time. Also, this is almost the best we can hope for [11]. There are more recent improvements to Chazelle’s result (see, e.g., Chan et al. [9]), but they are minor (at most a log factor), as the original results were already very close to matching the lower bound.

Most of these range search/counting problems can be reduced to the dominance range searching problem (on semigroups), where the query is represented by a point  $\mathbf{q}$ , and the objective is to return  $\bigoplus_{\mathbf{q} \leq \mathbf{p} \in S} w(\mathbf{p})$ . Here,  $\leq$  denotes the “dominance” relation (coordinate-wise  $\leq$ ). We can think of  $\mathbf{q}$  as the lower-corner of an infinite rectangle query.

## 3 RELAXED TREE DECOMPOSITIONS AND RELAXED POLYMATROIDS

### 3.1 Connection to a geometric data structure

We always assume that  $\bigcup_{S \in \mathcal{E}_s} S = \mathcal{V}$ ; otherwise the output size  $|Q|$  could be infinite. We start with a special case of (3) in which the skeleton part  $\mathcal{E}_s$  contains only *two* hyperedges  $U$  and  $L$ . Consider the aggregate query of the form

$$Q(\mathbf{x}_F) = \bigoplus_{\mathbf{x}_{\mathcal{V}_1 F}} \Phi_1(\mathbf{x}_U) \otimes \Phi_2(\mathbf{x}_L) \otimes \left( \bigotimes_{S \in \mathcal{E}_{\ell}} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right), \quad (14)$$

where  $\Phi_1$  and  $\Phi_2$  are two input functions/relations over variable sets  $U$  and  $L$ , respectively. We prove the following very simple but important lemma:

**Lemma 3.1.** *Let  $N = \max\{|\Phi_1|, |\Phi_2|\}$ , and  $k = |\mathcal{E}_{\ell}|$ , then when  $F \subseteq U$ , query (14) can be answered in time  $O(N \cdot (\log N)^{\max(k-1, 1)})$ .*

**PROOF.** If there is a hyperedge  $S \in \mathcal{E}_{\ell}$  for which  $S \subseteq U$ , then in a  $O(N \log N)$ -time pre-processing step we can “absorb” the factor  $\mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0}$  into the factor  $\Phi_1$ , by replacing  $\Phi_1(\mathbf{x}_U)$  with  $\Phi_1(\mathbf{x}_U) \otimes \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0}$ . A similar absorption

<sup>2</sup>In a semigroup we can add two elements using  $\oplus$ , but there is no additive inverse.

can be done with  $S \subseteq L$ . Hence, without loss of generality we can assume that  $S \not\subseteq L$  and  $S \not\subseteq U$  for all  $S \in \mathcal{E}_\ell$ . Furthermore, we only need to show that we can compute (14) for  $F = U$ , because after  $Q(\mathbf{x}_U)$  is computed, we can marginalize away variables  $\mathbf{x}_{U \setminus F}$  in  $O(N \log N)$ -time.

Abusing notation somewhat, for each  $S \in \mathcal{E}_\ell$  and each  $T \subseteq S$ , define the function  $\theta_T^S : \prod_{v \in T} \text{Dom}(X_v) \rightarrow \mathbb{R}$  by

$$\theta_T^S(\mathbf{x}_T) := \sum_{v \in T} \theta_v^S(x_v). \quad (15)$$

Fix a tuple  $\mathbf{x}_U$  such that  $\Phi_1(\mathbf{x}_U) \neq \mathbf{0}$ . A tuple  $\mathbf{x}_L$  is said to be  $\mathbf{x}_U$ -adjacent if  $\pi_{U \cap L} \mathbf{x}_U = \pi_{U \cap L} \mathbf{x}_L$ . We show how to compute the following sum in poly-logarithmic time:

$$\bigoplus_{\mathbf{x}_L \setminus U} \Phi_1(\mathbf{x}_U) \otimes \Phi_2(\mathbf{x}_L) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right) = \quad (16)$$

$$\Phi_1(\mathbf{x}_U) \otimes \bigoplus_{\mathbf{x}_L \setminus U} \Phi_2(\mathbf{x}_L) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\theta_{S \cap U}^S(\mathbf{x}_{S \cap U}) \leq -\theta_{S \setminus U}^S(\mathbf{x}_{S \setminus U})} \right). \quad (17)$$

where the inner sum ranges over only tuples  $\mathbf{x}_L$  which are  $\mathbf{x}_U$ -adjacent; non-adjacent tuples contribute  $\mathbf{0}$ .

Now, for the fixed  $\mathbf{x}_U$  and for each  $\mathbf{x}_L$  define the following  $k$ -dimensional points:

$$\mathbf{q}(\mathbf{x}_U) = (q_S(\mathbf{x}_U))_{S \in \mathcal{E}_\ell} \quad \text{where} \quad q_S(\mathbf{x}_U) := \theta_{S \cap U}^S(\mathbf{x}_{S \cap U}),$$

$$\mathbf{p}(\mathbf{x}_L) = (p_S(\mathbf{x}_L))_{S \in \mathcal{E}_\ell} \quad \text{where} \quad p_S(\mathbf{x}_L) := -\theta_{S \setminus U}^S(\mathbf{x}_{S \setminus U}).$$

We write  $\mathbf{q}(\mathbf{x}_U) \leq \mathbf{p}(\mathbf{x}_L)$  to say that  $\mathbf{q}(\mathbf{x}_U)$  is dominated by  $\mathbf{p}(\mathbf{x}_L)$  coordinate-wise:  $q_S(\mathbf{x}_U) \leq p_S(\mathbf{x}_L) \forall S \in \mathcal{E}_\ell$ . Assign to each point  $\mathbf{p}(\mathbf{x}_L)$  a “weight” of  $\Phi_2(\mathbf{x}_L)$ . Now, taking (17),

$$\begin{aligned} & \bigoplus_{\mathbf{x}_L \setminus U} \Phi_2(\mathbf{x}_L) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\theta_{S \cap U}^S(\mathbf{x}_{S \cap U}) \leq -\theta_{S \setminus U}^S(\mathbf{x}_{S \setminus U})} \right) \\ &= \bigoplus_{\mathbf{x}_L \setminus U} \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{q_S(\mathbf{x}_U) \leq p_S(\mathbf{x}_L)} \right) \otimes \Phi_2(\mathbf{x}_L) \quad (18) \\ &= \bigoplus_{\mathbf{x}_L \setminus U} \mathbf{1}_{\mathbf{q}(\mathbf{x}_U) \leq \mathbf{p}(\mathbf{x}_L)} \otimes \Phi_2(\mathbf{x}_L). \quad (19) \end{aligned}$$

The expression thus computes, for a given “query point”  $\mathbf{q}(\mathbf{x}_U)$ , the weighted sum over all points  $\mathbf{p}(\mathbf{x}_L)$  that dominate the query point. This is precisely the dominance range counting problem, which—modulo a  $O(N(\log N)^{\max(k-1,1)})$ -preprocessing step—can be solved in time  $O((\log N)^{\max(k-1,1)})$  [10], as reviewed in Section 2.3.

To conclude the proof, note that (14) can be written as (assuming  $F \subseteq U$  as is the case in Lemma 3.1)

$$Q(\mathbf{x}_F) = \bigoplus_{\mathbf{x}_{U \setminus F}} \underbrace{\bigoplus_{\mathbf{x}_L \setminus U} \Phi_1(\mathbf{x}_U) \otimes \Phi_2(\mathbf{x}_L) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right)}_{\text{same as (16)}},$$

where the outer sum ranges over  $N$  tuples  $\mathbf{x}_U$  in  $\Phi_1$ .  $\square$

**Example 3.2.** Let  $R$  be a binary relation. Suppose we want to count the number of tuples satisfying  $R(a, b) \wedge R(b, c) \wedge a < c$ , then by setting  $F = \emptyset$ ,  $U = \{a, b\}$ ,  $L = \{b, c\}$ , it is easy to see that the problem can be reduced to the form (14) with  $k = 1$ ,  $\mathcal{E}_\ell = \{\{a, c\}\}$ . We can thus compute this count in time  $O(N \log N)$ .  $\square$

### 3.2 Relaxed tree decompositions

Equipped with this basic case, we can now proceed to solve the general setting of (3). To this end, we define a new width parameter.

**Definition 3.3** (Relaxed tree decomposition). Let  $\mathcal{H} = (\mathcal{V}, \mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_\ell)$  denote a multi-hypergraph whose edge multiset is partitioned into  $\mathcal{E}_s$  and  $\mathcal{E}_\ell$ . A relaxed tree decomposition of  $\mathcal{H}$  (with respect to the partition  $\mathcal{E}_s \cup \mathcal{E}_\ell$ ) is a pair  $(T, \chi)$ , where  $T = (V(T), E(T))$  is a tree, and  $\chi : V(T) \rightarrow 2^{\mathcal{V}}$  satisfies the following properties:

- The running intersection property holds: for each node  $v \in \mathcal{V}$  the set  $\{t \in V(T) \mid v \in \chi(t)\}$  is a connected subtree in  $T$ .
- Every “skeleton” edge  $S \in \mathcal{E}_s$  is covered by some bag  $\chi(t)$ ,  $t \in V(T)$ .
- Every “ligament” edge  $S \in \mathcal{E}_\ell$  is covered by the union of two adjacent bags:  $S \subseteq \chi(s) \cup \chi(t)$ , where  $\{s, t\} \in E(T)$ .

Let  $\text{TD}^\ell(\mathcal{H})$  denote the set of all relaxed tree decompositions of  $\mathcal{H}$  (with respect to the skeleton-ligament partition). When  $\mathcal{H}$  is clear from context we use  $\text{TD}^\ell$  for the sake of brevity. Given  $F \subseteq \mathcal{V}$ , let  $\text{TD}_F^\ell$  denote the set of all relaxed  $F$ -connex tree decompositions of  $\mathcal{H}$ .

**3.2.1 FAQ-AI on a general semiring.** We use relaxed TDs in conjunction with Lemma 3.1 to answer FAQ-AI with a relaxed notion of faqw. In particular, the relaxed width parameters of  $\mathcal{H}$  are defined in exactly the same way as the usual width parameters defined in Section 2, except we allow the TDs to range over relaxed ones.

**Definition 3.4** (Relaxed faqw). Let  $Q$  be an FAQ-AI query (3), and  $\mathcal{H} = (\mathcal{V}, \mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_\ell)$  be its hypergraph. Furthermore, let  $\mathcal{E}_{\neq \emptyset} \subseteq \mathcal{E}_s$  denote the set of hyperedges  $K \in \mathcal{E}$  for which  $|R_K| < \infty$ . Then, the relaxed FAQ-width of  $Q$  is defined by

$$\text{faqw}_\ell(Q) := \min_{(T, \chi) \in \text{TD}_F^\ell} \max_{h \in \text{ED}_{\neq \emptyset} \cap \Gamma_n} \max_{t \in V(T)} h(\chi(t)) \quad (20)$$

When  $F = \emptyset$ ,  $\text{faqw}_\ell$  collapses to  $\text{fhtw}_\ell$ , in which case we define the relaxed fhtw for FAQ-AI  $Q$  without free variables:

$$\text{fhtw}_\ell(Q) := \min_{(T, \chi) \in \text{TD}_\emptyset^\ell} \max_{h \in \text{ED}_{\neq \emptyset} \cap \Gamma_n} \max_{t \in V(T)} h(\chi(t)) \quad (21)$$

A relaxed tree decomposition  $(T, \chi)$  of  $Q$  is *optimal* if its width is equal to  $\text{faqw}_\ell$ , i.e.,

$$\text{faqw}_\ell(Q) = \max_{h \in \text{ED}_{\mathcal{E}_\infty} \cap \Gamma_n} \max_{t \in V(T)} h(\chi(t)).$$

**Theorem 3.5.** Any FAQ-AI query  $Q$  of the form (3) on any semiring can be answered in time  $O(N^{\text{faqw}_\ell(Q)} \cdot (\log N)^{\max(k-1, 1) + |Q|})$ , where  $k$  is the maximum number of additive inequalities covered by a pair of adjacent bags in an optimal relaxed tree decomposition.<sup>3</sup>

**PROOF.** We first consider the case of no free variables because this case captures the key idea. Fix an optimal relaxed TD  $(T, \chi)$ . We first compute, for each bag  $t \in V(T)$  of the tree decomposition, a factor  $\Phi_t : \prod_{i \in \chi(t)} \text{Dom}(X_i) \rightarrow \mathcal{D}$  such that

$$\begin{aligned} Q() &= \bigoplus_{\mathbf{x}_V} \left( \bigotimes_{K \in \mathcal{E}_s} R_K(\mathbf{x}_K) \right) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right) \quad (22) \\ &= \bigoplus_{\mathbf{x}_V} \left( \bigotimes_{t \in V(T)} \Phi_t(\mathbf{x}_{\chi(t)}) \right) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right). \end{aligned} \quad (23)$$

To define the factors  $\Phi_t$ , we need the notion of *indicator projection* [5, 6]. For a given  $K \in \mathcal{E}_s$  and  $t \in V(T)$  such that  $J := K \cap \chi(t) \neq \emptyset$ , the indicator projection of  $R_K$  onto the bag  $\chi(t)$  is a function  $\pi_{t,K} : \prod_{v \in J} \text{Dom}(X_v) \rightarrow \{0, 1\}$  defined by

$$\pi_{t,K}(\mathbf{x}_J) := \begin{cases} 1 & \exists \mathbf{x}_{K \setminus J} \text{ s.t. } R_K((\mathbf{x}_J, \mathbf{x}_{K \setminus J})) \neq 0, \\ 0 & \text{otherwise.} \end{cases} \quad (24)$$

Recall from Definition 3.3 that every  $K \in \mathcal{E}_s$  is covered by at least one bag  $\chi(t)$  for  $t \in V(T)$ . Fix an arbitrary coverage assignment  $\alpha : \mathcal{E}_s \rightarrow V(T)$ , where  $K$  is covered by the bag  $\chi(\alpha(K))$ . Then, the factors  $\Phi_t$  are defined by:

$$\Phi_t(\mathbf{x}_{\chi(t)}) := \bigotimes_{K \in \alpha^{-1}(t)} R_K(\mathbf{x}_K) \otimes \bigotimes_{\substack{K \in \mathcal{E}_s \\ K \cap \chi(t) \neq \emptyset}} \pi_{t,K}(\mathbf{x}_{K \cap \chi(t)}). \quad (25)$$

It is easy to verify that (23) holds. Using a worst-case optimal join algorithm [28, 29, 34] we can compute (25) in time

$$\tilde{O}(N^{\rho_{\mathcal{E}_\infty}^*(\chi(t))}) = O(N^{\max_{h \in \text{ED}_{\mathcal{E}_\infty} \cap \Gamma_n} h(\chi(t))}). \quad (26)$$

Over all  $t \in V(T)$ , our runtime is bounded by  $O(N^w)$ , where

$$w = \max_{t \in V(T)} \max_{h \in \text{ED}_{\mathcal{E}_\infty} \cap \Gamma_n} h(\chi(t)). \quad (27)$$

The support of each factor  $\Phi_t$  has size bounded by  $N^w$ .

Next we compute (23) in time  $\tilde{O}(N^w)$ . We will make use of the fact that  $(T, \chi)$  is a relaxed TD. Fix an arbitrary root of the tree decomposition  $(T, \chi)$ ; following InsideOut, we compute (23) by eliminating variables from the leaves of  $(T, \chi)$  up to the root. Without loss of generality, we assume

<sup>3</sup>Note that  $k$  can be a lot smaller than  $|\mathcal{E}_\ell|$ .

that the tree decomposition is non-redundant, i.e., no bag is a subset of another in the tree decomposition (otherwise the contained bag factor can be “absorbed” into the containee bag factor). Let  $t_1$  be any leaf of  $(T, \chi)$ ,  $t_2$  be its parent, where  $L = \chi(t_1)$  and  $U = \chi(t_2)$ . Now write (23) as follows:

$$\begin{aligned} & \bigoplus_{\mathbf{x}_V} \left( \bigotimes_{t \in V(T)} \Phi_t(\mathbf{x}_{\chi(t)}) \right) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right) \\ &= \bigoplus_{\mathbf{x}_{V \setminus (L \cup U)}} \bigoplus_{\mathbf{x}_{L \cup U}} \left( \bigotimes_{t \in V(T)} \Phi_t(\mathbf{x}_{\chi(t)}) \right) \otimes \left( \bigotimes_{S \in \mathcal{E}_\ell} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right) \\ &= \bigoplus_{\mathbf{x}_{V \setminus (L \cup U)}} \left( \bigotimes_{t \in V(T) \setminus \{t_1, t_2\}} \Phi_t(\mathbf{x}_{\chi(t)}) \right) \otimes \left( \bigotimes_{\substack{S \in \mathcal{E}_\ell \\ S \cap (L \cup U) = \emptyset}} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right) \\ & \quad \otimes \left[ \bigoplus_{\mathbf{x}_{L \cup U}} \Phi_{t_1}(\mathbf{x}_L) \otimes \Phi_{t_2}(\mathbf{x}_U) \otimes \left( \bigotimes_{\substack{S \in \mathcal{E}_\ell \\ S \cap (L \cup U) \neq \emptyset}} \mathbf{1}_{\sum_{v \in S} \theta_v^S(x_v) \leq 0} \right) \right]. \end{aligned}$$

sub-query of the form (14)

The third equality uses the semiring’s distributive law. (Note that  $S \cap (L \cup U) \neq \emptyset$  implies that  $S \subseteq (L \cup U)$  thanks to Definition 3.3 and the fact that  $t_2$  is the only neighbor of  $t_1$ .) Lemma 3.1 implies that we can compute the sub-query in the allotted time. The above step eliminates all variables in  $L \cup U$ . Repeatedly applying the above step yields the desired output  $Q()$ .

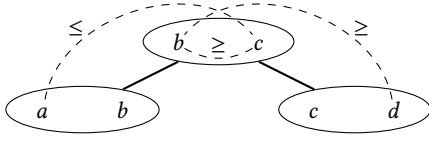
When the query has free variables, the algorithm proceeds similarly to the case of an FAQ with free variables [5].  $\square$

**Example 3.6.** Given 3 binary relations  $R, S$  and  $T$ , consider a query  $Q$  about the number of tuples  $(a, b, c, d)$  that satisfy:

$$R(a, b) \wedge S(b, c) \wedge T(c, d) \wedge (a \leq c) \wedge (c \leq b) \wedge (d \leq b). \quad (28)$$

The query  $Q$  has  $\mathcal{E}_s = \mathcal{E}_{\mathcal{E}_\infty} = \{\{a, b\}, \{b, c\}, \{c, d\}\}$  and  $\mathcal{E}_\ell = \mathcal{E}_\infty = \{\{a, c\}, \{b, c\}, \{b, d\}\}$ . Let  $N = \max\{|R|, |S|, |T|\}$ . Note that  $\text{faqw}(Q) = 2$ . In fact, any of the previously known algorithms, e.g. [5, 6], would take time  $O(N^2)$  to answer  $Q$ . However, this query has  $\text{faqw}_\ell(Q) = 1$ , and by Theorem 3.5, it can be answered in time  $O(N \cdot \log N)$ . (Note that here  $2 = k < |\mathcal{E}_\ell| = 3$ .) An optimal relaxed tree decomposition is shown in Figure 1.  $\square$

We next give a couple of simple lower and upper bounds for  $\text{faqw}_\ell$ . The upper bound shows that, effectively  $\text{faqw}_\ell$  is the best we can hope for, if the FAQ-AI query is arbitrary. The lower bound shows that, while the relaxed tree decomposition idea can improve the runtime by a polynomial factor, it cannot improve the runtime over straightforwardly applying InsideOut (over non-relaxed tree decompositions) by more than a polynomial factor.



**Figure 1: An optimal relaxed tree decomposition for the query in Example 3.6. Ligament edges are dashed. Each skeleton edge is held in one bag.**

**Proposition 3.7.** *For any positive integer  $m$ , there exists an FAQ-AI query of the form (3) for which  $F = \emptyset$ ,  $\text{faqw}_\ell(Q) \geq m$  and it cannot be answered in time  $o(N^{\text{faqw}_\ell(Q)})$ , modulo  $k$ -sum hardness.*

**Proposition 3.8.** *For any FAQ-AI query  $Q$  of the form (3), we have  $\text{faqw}_\ell(Q) \geq \frac{1}{2}\text{faqw}(Q)$ ; in particular, when  $Q$  has no free variables  $\text{fhtw}_\ell(Q) \geq \frac{1}{2}\text{fhtw}(Q)$ .*

**3.2.2 FAQ-AI on the Boolean semiring.** Before explaining how we can adapt PANDA to solve an FAQ-AI query on the Boolean semiring, we give the intuition with an example.

**Example 3.9.** Consider the following FAQ-AI:

$$Q() \leftarrow R(a, b) \wedge S(b, c) \wedge T(c, d) \wedge a \leq d. \quad (29)$$

Here  $\text{faqw}_\ell(Q) = \text{faqw}(Q) = 2$ . Using fractional hypertree width measure and InsideOut (even with relaxed TDs and Theorem 3.5), the best runtime is  $O(N^2)$ , because no matter which (relaxed) TD we choose, the worst-case bag relation size is  $\Theta(N^2)$ . A key idea of the PANDA framework [7] is the use of a *disjunctive Datalog rule*. Consider the following disjunctive Datalog rule:

$$U(a, b, c) \vee W(b, c, d) \leftarrow R(a, b) \wedge S(b, c) \wedge T(c, d). \quad (30)$$

There are two relations in the head  $U$  and  $W$ , and they form a solution to the rule iff the following holds: if  $(a, b, c, d)$  satisfies the body, then either  $(a, b, c) \in U$  or  $(b, c, d) \in W$ . Via information-theoretic inequalities [7], we are able to show that PANDA can compute a solution  $(U, W)$  to the above disjunctive Datalog rule in time  $\tilde{O}(N^{1.5})$ . In particular, both  $|U|$  and  $|W|$  are bounded by  $N^{1.5}$ .

Given the solution  $(U, W)$  to (30), it is straightforward to verify that the following also holds, using the distributivity of  $\vee$  over  $\wedge$ :

$$\begin{aligned} (R(a, b) \wedge W(b, c, d)) \vee (U(a, b, c) \wedge T(c, d)) \\ \leftarrow R(a, b) \wedge S(b, c) \wedge T(c, d). \end{aligned} \quad (31)$$

By semijoin-reducing  $W$  against  $S, T$ , and semijoin-reducing  $U$  against  $R, S$ , we conclude that

$$\begin{aligned} (R(a, b) \wedge W(b, c, d)) \vee (U(a, b, c) \wedge T(c, d)) \equiv \\ R(a, b) \wedge S(b, c) \wedge T(c, d). \end{aligned}$$

Finally, we have a rewrite of the original body:

$$\begin{aligned} [R(a, b) \wedge W(b, c, d) \wedge a \leq d] \vee [U(a, b, c) \wedge T(c, d) \wedge a \leq d] \\ \equiv R(a, b) \wedge S(b, c) \wedge T(c, d) \wedge a \leq d. \end{aligned} \quad (32)$$

By defining intermediate rules, we can compute  $Q$  from them:

$$Q_1() \leftarrow R(a, b) \wedge W(b, c, d) \wedge a \leq d, \quad (33)$$

$$Q_2() \leftarrow U(a, b, c) \wedge T(c, d) \wedge a \leq d, \quad (34)$$

$$Q() \leftarrow Q_1() \vee Q_2(). \quad (35)$$

$Q_1$  and  $Q_2$  are of the form (14), and thus they each can be answered in  $\tilde{O}(N^{1.5})$ -time (since  $|U|, |W| \leq N^{1.5}$ ). This implies that  $Q$  can be answered in  $\tilde{O}(N^{1.5})$ -time overall.  $\square$

The strategy outlined in the above example uses PANDA to evaluate an FAQ-AI query over the Boolean semiring. The resulting algorithm achieves a natural generalization of the submodular FAQ-width defined in (13):

**Definition 3.10.** Given an FAQ-AI query  $Q$  (3) over the Boolean semiring. The *relaxed submodular FAQ-width* of  $Q$  is defined by

$$\text{smfw}_\ell(Q) := \max_{h \in \text{ED}_{\neq} \cap \Gamma_n} \min_{(T, \chi) \in \text{TD}_F^\ell} \max_{t \in V(T)} h(\chi(t)). \quad (36)$$

(Recall that the set of relaxed tree decompositions  $\text{TD}_F^\ell$  was defined in Definition 3.3.)

**Theorem 3.11.** *Any FAQ-AI query  $Q$  of the form (3) on the Boolean semiring can be answered in time  $\tilde{O}(N^{\text{smfw}_\ell(Q)} + |Q|)$ .*

The proof of Theorem 3.11 can be found in Appendix A.1.

### 3.3 Relaxed polymatroids

A key step in the proof of Theorem 3.11 is to find the Boolean tensor decomposition (72) of the product over  $R_K$ . In a non-Boolean semiring, this becomes a tensor decomposition on this semiring:

$$\bigotimes_{K \in \mathcal{E}_s} R_K = \bigoplus_{(T, \chi) \in \text{TD}_F^\ell} \bigotimes_{t \in V(T)} S_{\chi(t)}^{(T, \chi)}. \quad (37)$$

In order to compute this tensor decomposition, we can still follow the script of the proof of Theorem 3.11, working on the parameter space of the input factors  $R_K$ ; however, for the equality in (37) to hold (it is an identity over the value-space of the factors), we have to ensure the following property:

For any  $\mathbf{x}_V$  s.t.  $\bigotimes_{K \in \mathcal{E}_s} R_K(\mathbf{x}_K) \neq \mathbf{0}$ , there is *exactly one* tree decomposition  $(T, \chi) \in \text{TD}_F^\ell$  for which

$$\bigotimes_{t \in V(T)} S_{\chi(t)}^{(T, \chi)}(\mathbf{x}_{\chi(t)}) = \bigotimes_{K \in \mathcal{E}_s} R_K(\mathbf{x}_K), \quad (38)$$

while for the other TDs, the left-hand side above is  $\mathbf{0}$ .



Essentially, the property ensures that we do not have to perform inclusion-exclusion (IE) over the tree decompositions in  $\text{TD}_F^\ell$ .<sup>4</sup> We do not know how to ensure this property in general. However, under a relaxed notion of polymatroids, the property above holds. Since this idea applies to FAQ queries in general, we start with our result on FAQ queries first, before specializing it to FAQ-AI.

**3.3.1 FAQ over an arbitrary semiring.** To explain how we can guarantee the property (38) for an FAQ query over an arbitrary semiring, consider the following example. Suppose that we would like to evaluate the (aggregate) query

$$Q() = \sum_{x_{[4]}} R_{12}(x_1, x_2) R_{23}(x_2, x_3) R_{34}(x_3, x_4) R_{41}(x_4, x_1). \quad (39)$$

We write  $R_{ij}$  instead of  $R_{ij}(x_i, x_j)$  for short. The factors  $R_{ij}$  are functions of two variables  $R_{ij} : \text{Dom}(X_i) \times \text{Dom}(X_j) \rightarrow \mathbb{R}$ , and they are represented by *ternary* relations in a database. Abusing notation we will also use  $R_{ij}$  to refer to its support, i.e., the binary relation over  $(X_i, X_j)$  such that  $(x_i, x_j) \in R_{ij}$  iff  $R_{ij}(x_i, x_j) \neq 0$ .

There are only two non-trivial tree decompositions for the “4-cycle” query (39): one with bags  $\{1, 2, 3\}$  and  $\{3, 4, 1\}$ , and the other with bags  $\{1, 2, 4\}$  and  $\{2, 3, 4\}$ .<sup>5</sup> To evaluate the query, we first solve the relation equation (37), but only on the supports; i.e., we would like to find relations  $S_{123}, S_{341}, S_{234}$ , and  $S_{412}$  such that

$$R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41} \equiv (S_{123} \wedge S_{341}) \vee (S_{234} \wedge S_{412}) \equiv (S_{123} \vee S_{234}) \wedge (S_{123} \vee S_{412}) \wedge (S_{341} \vee S_{234}) \wedge (S_{341} \vee S_{412}). \quad (40)$$

The second  $\equiv$  is due to the distributivity of  $\vee$  over  $\wedge$ . Since the last formula is in CNF, we can solve each term separately by solving 4 different disjunctive Datalog rules:

$$(S_{123} \vee S_{234}) \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}, \quad (41)$$

$$(S_{123} \vee S_{412}) \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}, \quad (42)$$

$$(S_{341} \vee S_{234}) \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}, \quad (43)$$

$$(S_{341} \vee S_{412}) \leftarrow R_{12} \wedge R_{23} \wedge R_{34} \wedge R_{41}. \quad (44)$$

Applying the proof-to-algorithm conversion idea from [7], the above disjunctive Datalog rules can be solved with the PANDA algorithm. It is beyond the scope of the main body of this paper to describe the PANDA algorithm in full details. However, we can describe a solution. Let  $N = \max\{|R_{12}|, |R_{23}|, |R_{34}|, |R_{41}|\}$ . For each input relation/factor, define their “light”

<sup>4</sup>IE is difficult for two reasons: (1) IE computation explodes the runtime, and (2) in a general semiring there may not be additive inverses and thus IE may not even apply.

<sup>5</sup>The trivial TD with one bag  $\{1, 2, 3, 4\}$  can always be replaced by a non-trivial TD in the considered bounds/algorithms without making them any worse. Similarly, redundant TDs can be replaced by non-redundant ones.

parts as follows.

$$R_{ij}^\ell := \{(x_i, x_j) \in R_{ij} : |\sigma_{X_i=x_i} R_{ij}| \leq \sqrt{N}\}. \quad (45)$$

Also, for every  $R_{ij}$ , define  $R_{ij}^h := R_{ij} \setminus R_{ij}^\ell$ . Then, one can verify that the following is a solution to the relational equations (41)–(44) (and by semijoin-reducing each one of them with relations  $R_{ij}$ , they become a solution to (40) as well):

$$S_{ijk} = R_{ij} \bowtie R_{jk}^l \cup \pi_i R_{ij}^h \bowtie R_{jk}. \quad (46)$$

Furthermore, it is straightforward to verify that each  $S_{ijk}$  can be computed in  $\tilde{O}(N^{1.5})$ -time. Once we have obtained this solution to the relational equation, i.e., we have the relations  $S_{ijk}$ , we can extend them naturally into factors (so that they are represented by 4-ary relations) satisfying (38). In particular, as functions with range  $\mathbb{R}$ , they are defined by

$$S_{ijk}(x_i, x_j, x_k) := R_{ij}(x_i, x_j) \cdot R_{jk}(x_j, x_k). \quad (47)$$

The above sketch does not work for a generic FAQ query because we do not know how to guarantee that (38) is satisfied given the relational solution returned by PANDA. (If we were able to do so, then the notion of submodular width would apply also to #CSP and not just CSP.) However, we *are* able to prove that this strategy works (i.e., (38) can be ensured) under a relaxed notion of polymatroids and a corresponding “sharp submodular (FAQ) width”.

**Definition 3.12.** Given a collection  $\mathcal{E}$  of subsets of  $\mathcal{V}$ , a set function  $h : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$  is said to be a  $\mathcal{E}$ -*polymatroid* if it satisfies the following: (i)  $h(\emptyset) = 0$ , (ii)  $h(X) \leq h(Y)$  whenever  $X \subseteq Y$ , and (iii)  $h(X \cup Y) + h(X \cap Y) \leq h(X) + h(Y)$  for every pair  $X, Y \subseteq \mathcal{V}$  such that  $X \cap Y \subseteq S$  for some  $S \in \mathcal{E}$ . In particular, a  $2^{\mathcal{V}}$ -polymatroid is a polymatroid as defined in Section 2.1. For  $\mathcal{V} = [n]$ , let  $\Gamma_{n|\mathcal{E}}$  denote the set of all  $\mathcal{E}$ -polymatroids on  $\mathcal{V}$ .

The following definition is a straightforward generalization of smfw from (13), where we replace  $\Gamma_n$  by the relaxed polymatroids.

**Definition 3.13.** Given an FAQ query (2) whose hypergraph is  $\mathcal{H} = (\mathcal{V}, \mathcal{E} = \mathcal{E}_{\neq \emptyset} \cup \mathcal{E}_{\infty})$ , its *#-submodular FAQ-width*, denoted by  $\#\text{smfw}(Q)$ , is defined by

$$\#\text{smfw}(Q) := \max_{h \in \text{ED}_{\neq \emptyset} \cap \Gamma_{n|\mathcal{E}_{\neq \emptyset}}} \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} h(\chi(t)). \quad (48)$$

When there are no free variables, i.e.,  $F = \emptyset$ , we define  $\#\text{subw}(Q) := \#\text{smfw}(Q)$ , to mirror the case when  $\text{faqw}(Q) = \text{fhtw}(Q)$ .

Under the above more restricted width parameter<sup>6</sup>, our vision above with condition (38) can now be realized:

<sup>6</sup>When we relax the polymatroids, the width goes *up*, and thus it is more restricted.

**Theorem 3.14.** Any FAQ query  $Q$  of the form (2) on any semiring can be answered in time  $\tilde{O}(N^{\#\text{smfw}(Q)} + |Q|)$ .

Appendix A.2 gives the proof of Theorem 3.14, which involves an appropriate adaptation of PANDA called #PANDA. Appendix A.3 shows that while  $\#\text{smfw}(Q)$  can be larger than  $\text{smfw}(Q)$ , it is not larger than  $\text{faqw}(Q)$  and can be unboundedly smaller for classes of queries.

**Proposition 3.15** (Connecting #smfw to smfw and faqw).

(a) For any FAQ query  $Q$ , the following holds:

$$\text{smfw}(Q) \leq \#\text{smfw}(Q) \leq \text{faqw}(Q). \quad (49)$$

In particular, when  $Q$  has no free variables, we have

$$\text{subw}(Q) \leq \#\text{subw}(Q) \leq \text{fhtw}(Q). \quad (50)$$

(b) Furthermore, there are classes of queries  $Q$  for which the gap between  $\#\text{smfw}(Q)$  and  $\text{faqw}(Q)$  is unbounded, and so is the gap between  $\#\text{subw}(Q)$  and  $\text{fhtw}(Q)$ .

**Example 3.16.** Consider again the count query  $Q$  in (39), which we showed earlier how to compute in time  $\tilde{O}(N^{1.5})$ . Since  $Q$  has no free variables,  $\text{faqw}(Q) = \text{fhtw}(Q) = 2$  and  $\#\text{smfw}(Q) = \#\text{subw}(Q)$ . In the proof of Proposition 3.15, we show that  $\#\text{subw}(Q) \leq 1.5$ . Therefore, the #PANDA algorithm from the proof of Theorem 3.14 can compute (39) in time  $\tilde{O}(N^{1.5})$ . In fact, the  $\tilde{O}(N^{1.5})$  algorithm we described earlier for (39) is just a specialization of #PANDA. The proof of Proposition 3.15 offers a family of similar examples.  $\square$

**3.3.2 FAQ-AI over an arbitrary semiring.** Finally, we put everything together to solve the FAQ-AI problem. The only (very natural) change is to replace the tree decompositions by their relaxed version, and the technical details flow through.

**Definition 3.17.** Given an FAQ-AI query (3) whose hypergraph is  $\mathcal{H} = (\mathcal{V}, \mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_\ell = \mathcal{E}_{\neq} \cup \mathcal{E}_\infty)$ , its *relaxed #submodular FAQ-width*, denoted by  $\#\text{smfw}_\ell(Q)$ , is defined by

$$\#\text{smfw}_\ell(Q) := \max_{h \in \text{ED}_{\neq} \cap \Gamma_{\neq} \mathcal{E}_{\neq}} \min_{(T, \chi) \in \text{TD}_F^\ell} \max_{t \in V(T)} h(\chi(t)). \quad (51)$$

When  $F = \emptyset$ , we define  $\#\text{subw}_\ell(Q) := \#\text{smfw}_\ell(Q)$ .

**Theorem 3.18.** Any FAQ-AI query  $Q$  of the form (3) on any semiring can be computed in time  $\tilde{O}(N^{\#\text{smfw}_\ell(Q)} + |Q|)$ .

**Example 3.19.** Consider the following count query:

$$Q() = \sum_{a,b,c,d} R(a,b) \cdot S(b,c) \cdot T(c,d) \cdot \mathbf{1}_{a+b+c+d \leq 0}. \quad (52)$$

Let  $N := \max\{|R|, |S|, |T|\}$ . For the above query  $\text{faqw}(Q) = \text{faqw}_\ell(Q) = \#\text{smfw}(Q) = 2$ . Any of the previously known algorithms, including the one from Theorem 3.5 and the one from Theorem 3.14, would need time  $O(N^2)$  to compute  $Q$ . In Appendix A.4, we show that  $\#\text{smfw}_\ell(Q) \leq 1.5$ . As an example of Theorem 3.18, we also show how to compute the

above query in  $\tilde{O}(N^{1.5})$ . (Using the same method, we can also solve the counting version of  $Q_3$  from Example 1.2 in the same time.)  $\square$

## 4 APPLICATIONS TO RELATIONAL MACHINE LEARNING

Our FAQ-AI formalism and solution are directly applicable to learning a class of machine learning models, which includes supervised models (e.g., robust regression, SVM classification), and unsupervised models (e.g., clustering via  $k$ -means). In this section, we show that the core computation of these optimization problems can be formulated in FAQ-AI over the sum-product semiring.

### 4.1 Training ML models over databases

A typical machine learning model is learned over a training dataset  $G$ . We consider the common scenario where the input data is a relational database  $I$ , and the training dataset  $G$  is the result of a feature extraction join query  $Q$  over  $I$  [2, 3, 18, 24]. Each tuple  $(\mathbf{x}, y) \in G$  consists of a vector of features  $\mathbf{x}$  and a label  $y$ . We consider that the feature extraction query  $Q$  has the hypergraph  $\mathcal{H} = (\mathcal{V}, \mathcal{E}_s)$ , where  $\mathcal{E}_s$  is the set of its skeleton hyperedges.

A supervised machine learning model is a function  $f_\beta(\mathbf{x})$  with parameters  $\beta$  that is used to predict the label  $y$  for unlabeled data. The parameters are obtained by minimizing the objective function:

$$J(\beta) = \sum_{(\mathbf{x}, y) \in G} \mathcal{L}(y, f_\beta(\mathbf{x})) + \lambda \Omega(\beta), \quad (53)$$

where  $\mathcal{L}(a, b)$  is a loss function,  $\Omega$  is a regularizer, e.g.,  $\ell_1$  or  $\ell_2$  norm, and the constant  $\lambda \in (0, 1)$  controls the influence of regularization.

Previous work has shown that for polynomial loss functions, such as square loss  $\mathcal{L}(a, b) = (a - b)^2$ , the core computation for optimizing the objective  $J(\beta)$  amounts to FAQ evaluation [3]. In many instances, however, the loss function is non-polynomial, either due to the structure of the loss, or the presence of non-polynomial components embedded within the model structure (e.g., ReLU activation function in neural nets) [27].

Examples of commonly used non-polynomial loss functions are: (1) hinge loss, used to learn classification models like linear support vector machines (SVM) [27], or generalized low rank models (glrm) with boolean principal component analysis (PCA) [33]; (2) Huber loss, used to learn regression models that are robust to outliers [27]; (3) scalene loss, used to learn quantile regression models [33]; (4) epsilon insensitive loss, used to learn SVM regression models [27]; and (5) ordinal hinge loss, used to learn ordinal regression models or ordinal PCA (another glrm) [33].

Any optimization problem with the above non-polynomial loss functions can benefit from our evaluation algorithm for FAQ-AI by reformulating computations in the optimization algorithm as FAQ-AI expressions over the feature extraction join query  $Q$ . We next exemplify this reformulation for two such problems: (1) learning a robust linear regression model using Huber loss, which can be solved with gradient-descent optimization, and (2) learning a linear support vector machine (SVM) for binary classification using hinge loss, which can be solved with subgradient-based optimization algorithms or with a cutting-plane algorithm for the primal formulation of linear SVM classification. The full version [1] details the cases of the scalene, epsilon insensitive, and ordinal hinge loss functions.

We also consider  $k$ -means unsupervised clustering and give an FAQ-AI reformulation of the computation done in an iteration of the algorithm over the dataset  $G$ .

The advantage of FAQ-AI reformulation is that the FAQ-AI expressions for the aforementioned optimization problems can be evaluated over relaxed tree decompositions of the feature extraction query  $Q$  and do not require the explicit materialization of its result  $G$ . The size of and time to compute  $G$  is  $O(|I|^{\rho^*(Q)})$  [29]. The solution to these optimization problems can be computed in time sub-linear in the size of  $G$ , using InsideOut or #PANDA.

## 4.2 Robust linear regression with Huber loss

A linear regression model is a linear function  $f_{\beta}(\mathbf{x}) = \beta^{\top} \mathbf{x} = \sum_{i \in [n]} \beta_i x_i$  with features  $\mathbf{x} = (x_1 = 1, x_2, \dots, x_n)$  and parameters  $\beta = (\beta_1, \dots, \beta_n)$ . For a given feature vector  $\mathbf{x}$ , the model is used to estimate the (continuous) label  $y \in \mathbb{R}$ . We learn the model parameters by minimizing the objective  $J(\beta)$  with the Huber loss function, which is defined as:

$$\mathcal{L}(a, b) = \begin{cases} \frac{1}{2}(a - b)^2 & \text{if } |a - b| \leq 1, \\ \frac{1}{2}|a - b| - \frac{1}{2} & \text{otherwise.} \end{cases} \quad (54)$$

Huber loss is equivalent to the square loss when  $|a - b| \leq 1$  and to the absolute loss otherwise<sup>7</sup>. In contrast to the absolute loss, Huber loss is differentiable at all points. It is also more robust to outliers than the square loss.

To learn the parameters, we use batch gradient descent optimization, which repeatedly updates the parameters in the direction of the gradient  $\nabla J(\beta)$  until convergence. We provide details on gradient-based optimization in [1]. In this section, we focus on the core computation of the algorithm, which is the repeated computation of the objective  $J(\beta)$  and its gradient  $\nabla J(\beta)$ .

<sup>7</sup>Without loss of generality, we use a simplified Huber loss. The threshold between absolute and square loss is given by a constant  $\delta$  and the absolute loss is  $\frac{\delta}{2}|a - b| - \frac{\delta^2}{2}$ .

The gradient  $\nabla J(\beta)$  is the vector of partial derivatives with respect to parameters  $(\beta_j)_{j \in [n]}$ . The objective function  $J(\beta)$  (with  $\ell_2$  regularization) and its partial derivative with respect to  $\beta_j$  are:

$$J(\beta) = \frac{1}{2} \sum_{(x, y) \in G} \left[ (y - f_{\beta}(\mathbf{x}))^2 \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| \leq 1} + (|y - f_{\beta}(\mathbf{x})| - 1) \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| > 1} \right] + \frac{\lambda}{2} \|\beta\|_2^2, \quad (55)$$

$$\begin{aligned} \frac{\partial J(\beta)}{\partial \beta_j} &= \lambda \beta_j - \sum_{(x, y) \in G} \left[ (y - f_{\beta}(\mathbf{x})) \cdot x_j \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| \leq 1} \right. \\ &\quad \left. + \frac{1}{2}(x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) > 0} - x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) < 0}) \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| > 1} \right] \\ &= \lambda \beta_j - \sum_{(x, y) \in G} (y - f_{\beta}(\mathbf{x})) \cdot x_j \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| \leq 1} \\ &\quad - 1/2 \sum_{(x, y) \in G} x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) > 1} + 1/2 \sum_{(x, y) \in G} x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) < -1}. \end{aligned} \quad (56)$$

Our observation is that we can compute  $J(\beta)$  and  $\frac{\partial J(\beta)}{\partial \beta_j}$  without materializing  $G$ , by reformulating their data dependent computation as a few FAQ-AI expressions. We exemplify the rewriting for  $\frac{\partial J(\beta)}{\partial \beta_j}$ ; the rewriting for  $J(\beta)$  is presented in [1]. We rewrite the first of the three summations in  $\frac{\partial J(\beta)}{\partial \beta_j}$  as follows:

$$\begin{aligned} &\sum_{(x, y) \in G} (y - \sum_{i \in [n]} \beta_i x_i) \cdot x_j \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| \leq 1} \quad (57) \\ &= \sum_{(x, y) \in G} y \cdot x_j \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| \leq 1} - \sum_{i \in [n]} \sum_{(x, y) \in G} \beta_i \cdot x_i \cdot x_j \cdot \mathbf{1}_{|y - f_{\beta}(\mathbf{x})| \leq 1} \\ &= \sum_{(x, y) \in G} y \cdot x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) \leq 1} \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) > 0} \\ &\quad + \sum_{(x, y) \in G} y \cdot x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) \geq -1} \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) < 0} \\ &\quad - \sum_{i \in [n]} \sum_{(x, y) \in G} \beta_i \cdot x_i \cdot x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) \leq 1} \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) > 0} \\ &\quad - \sum_{i \in [n]} \sum_{(x, y) \in G} \beta_i \cdot x_i \cdot x_j \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) \geq -1} \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) < 0}. \end{aligned}$$

The four terms can be expressed as  $O(n)$  FAQ-AI expressions of the form (3). For instance, the first part of the expression is equivalent to the following FAQ-AI query:

$$Q() = \sum_{y, \mathbf{x}_V} y \cdot x_j \cdot \underbrace{\mathbf{1}_{y - f_{\beta}(\mathbf{x}) \leq 1} \cdot \mathbf{1}_{y - f_{\beta}(\mathbf{x}) > 0}}_{\text{ligaments } \mathcal{E}_{\ell}} \cdot \left( \prod_{F \in \mathcal{E}_s} R_F(\mathbf{x}_F) \right).$$

The other two summations in  $\frac{\partial J(\beta)}{\partial \beta_j}$  both aggregate over  $x_j$  and have one inequality that defines a ligament in  $\mathcal{E}_{\ell}$ . They can be expressed as FAQ-AI expressions. Overall, the gradient

$\nabla J(\boldsymbol{\beta})$  can be expressed as  $O(n^2)$  FAQ-AI expressions. The full version [1] shows that the same holds for  $J(\boldsymbol{\beta})$ .

**Theorem 4.1.** *Let  $I$  be an input database where  $N$  is the largest relation in  $I$ , and  $Q$  be a feature extraction query. For any robust linear regression model  $\boldsymbol{\beta}^\top \mathbf{x}$ , the objective  $J(\boldsymbol{\beta})$  and gradient  $\nabla J(\boldsymbol{\beta})$  with Huber loss can be computed in time  $\tilde{O}(N^{\#\text{smfw}_\ell(Q)})$  with #PANDA and in time  $O(N^{\text{faqw}_\ell(Q)} \log N)$  with InsideOut.*

### 4.3 Linear support vector machines

A linear SVM classification model is used for binary classification problems where the label  $y \in \{\pm 1\}$ . For the features  $\mathbf{x} = (x_1 = 1, x_2, \dots, x_n)$ , the model learns the parameters  $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$  of a linear discriminant function  $f_\beta(\mathbf{x}) = \boldsymbol{\beta}^\top \mathbf{x}$  such that  $f_\beta(\mathbf{x})$  separates the data points in  $G$  into positive and negative classes with a maximum margin. The parameters can be learned by minimizing the objective function (53) with the hinge loss function:

$$\mathcal{L}(a, b) = \max\{0, 1 - a \cdot b\}. \quad (58)$$

Hinge loss is non-differentiable, and thus standard gradient descent optimization is not applicable. We next discuss two alternative approaches for solving this optimization.

The first approach is based on the observation that the loss function is convex, and the objective admits subgradient vectors, which generalize the standard notion of gradient. The optimization problem can be solved with subgradient based updates. Pegasos is a well-know algorithm for this approach [32].

The alternative approach is to solve the primal formulation of the problem, which avoids the non-differentiable objective by turning it into a constraint optimization problem with slack variables. Joachims proposed a cutting-plane algorithm which solves this optimization problem efficiently [20].

For both approaches, the number of iterations of the optimization algorithm is independent of the size  $|G|$  of training dataset  $G$  [20, 32]. Thus, the time complexity for finding the solution is  $O(|G|)$ .

Despite the fact that the two approaches solve the same problem, they have been hugely influential in their own right. We therefore consider both approaches, and show that by reformulating their computation as FAQ-AI we can solve them asymptotically faster than materializing the training dataset  $G$ , i.e., sublinear in  $|G|$ .

**4.3.1 Subgradient-based optimization for linear SVM classification.** We first use subgradient-based optimization to compute the parameters of the SVM model; The full version [1] gives the details. The core of the optimization is the repeated computation of the objective and the partial derivatives in terms of  $(\beta_j)_{j \in [n]}$ . The objective  $J(\boldsymbol{\beta})$  (with  $\ell_2$

regularization) and the partial derivative  $\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$  are:

$$J(\boldsymbol{\beta}) = \sum_{(x,y) \in G} \max\{0, 1 - y(\boldsymbol{\beta}^\top \mathbf{x})\} + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2, \quad (59)$$

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j} = - \sum_{(x,y) \in G} y \cdot x_j \cdot \mathbf{1}_{y(\boldsymbol{\beta}^\top \mathbf{x}) \leq 1} + \lambda \beta_j. \quad (60)$$

Both  $J(\boldsymbol{\beta})$  and  $\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$  can be reformulated as FAQ-AI expressions and computed without materializing  $G$ . We first rewrite the objective (derivation steps shown in [1]):

$$\begin{aligned} \sum_{(x,y) \in G} \max\{0, 1 - y(\boldsymbol{\beta}^\top \mathbf{x})\} + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 & \quad (61) \\ = \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 + \underbrace{\sum_{(x,y) \in G} \mathbf{1}_{y=1} \mathbf{1}_{\boldsymbol{\beta}^\top \mathbf{x} \leq 1}}}_{\text{FAQ-AI of the form (3)}} - \underbrace{\sum_{i=1}^n \sum_{(x,y) \in G} \beta_i x_i \mathbf{1}_{y=1} \mathbf{1}_{\boldsymbol{\beta}^\top \mathbf{x} \leq 1}}}_{\text{FAQ-AI of the form (3)}} \\ & + \underbrace{\sum_{(x,y) \in G} \mathbf{1}_{y=-1} \mathbf{1}_{\boldsymbol{\beta}^\top \mathbf{x} \geq -1}}}_{\text{FAQ-AI of the form (3)}} + \underbrace{\sum_{i=1}^n \sum_{(x,y) \in G} \beta_i x_i \mathbf{1}_{y=-1} \mathbf{1}_{\boldsymbol{\beta}^\top \mathbf{x} \geq -1}}}_{\text{FAQ-AI of the form (3)}}. \end{aligned}$$

$\frac{\partial J(\boldsymbol{\beta})}{\partial \beta_j}$  can also be rewritten into two FAQ-AI expressions:

$$\begin{aligned} - \sum_{(x,y) \in G} y \cdot x_j \cdot \mathbf{1}_{y(\boldsymbol{\beta}^\top \mathbf{x}) \leq 1} + \lambda \beta_j & \quad (62) \\ = \lambda \beta_j - \underbrace{\sum_{(x,y) \in G} x_j \cdot \mathbf{1}_{y=1} \mathbf{1}_{\boldsymbol{\beta}^\top \mathbf{x} \leq 1}}}_{\text{FAQ-AI of the form (3)}} + \underbrace{\sum_{(x,y) \in G} x_j \cdot \mathbf{1}_{y=-1} \mathbf{1}_{\boldsymbol{\beta}^\top \mathbf{x} \geq -1}}}_{\text{FAQ-AI of the form (3)}}. \end{aligned}$$

**Theorem 4.2.** *Let  $I$  be an input database where  $N$  is the largest relation in  $I$ , and  $Q$  be a feature extraction query. For any linear SVM classification model  $\boldsymbol{\beta}^\top \mathbf{x}$ , the objective  $J(\boldsymbol{\beta})$  and gradient  $\nabla J(\boldsymbol{\beta})$  with hinge loss can be computed in time  $\tilde{O}(N^{\#\text{smfw}_\ell(Q)})$  with #PANDA and in time  $O(N^{\text{faqw}_\ell(Q)} \log N)$  with InsideOut.*

**4.3.2 Cutting-plane algorithm for linear SVM classification in primal space.** An alternative to learning linear SVM via subgradient-based optimization is to pose the problem as a constraint optimization problem. The equivalent formulation for minimizing the objective (59) is the primal formulation of linear SVM [27]:

$$\begin{aligned} \min_{\boldsymbol{\beta}, \xi_{x,y} \geq 0} \quad & \frac{1}{2} \|\boldsymbol{\beta}\|_2^2 + \frac{C}{|G|} \sum_{(x,y) \in G} \xi_{x,y} & (63) \\ \text{s.t.} \quad & y f_\beta(\mathbf{x}) \geq 1 - \xi_{x,y}, \quad \forall (x,y) \in G. \end{aligned}$$

where  $\xi_{x,y}$  are slack variables and  $C$  is the regularization parameter.

The optimization problem solves for the hyperplane  $f_\beta(\mathbf{x})$  that classifies the data points  $(\mathbf{x}, y) \in G$  into two classes, so that the margin between the hyperplane and the nearest data

**Algorithm 1:** Training classification SVM via (64)

---

```

1  $\mathcal{W} \leftarrow \emptyset;$  // Working set
2  $t \leftarrow 0;$ 
3 repeat
4    $t \leftarrow t + 1;$ 
5    $(\beta^{(t)}, \xi^{(t)}) \leftarrow \arg \min_{\beta, \xi \geq 0} \left\{ \frac{1}{2} \|\beta\|_2^2 + C\xi; \right.$ 
6      $\left. \text{s.t. } \frac{1}{|G|} \left\langle \beta, \sum_{(x,y) \in T} yx \right\rangle \geq \frac{|T|}{|G|} - \xi, \forall T \in \mathcal{W}; \right.$ 
7      $T^{(t)} := \{(x, y) \in G \mid y \langle \beta^{(t)}, x \rangle < 1\};$ 
8      $\mathcal{W} \leftarrow \mathcal{W} \cup \{T^{(t)}\}$ 
9 until  $\frac{|T^{(t)}|}{|G|} - \frac{1}{|G|} \left\langle \beta^{(t)}, \sum_{(x,y) \in T^{(t)}} yx \right\rangle \leq \xi^{(t)} + \epsilon;$ 

```

---

point for each class is maximized. For each  $(x, y) \in G$ , the slack variable  $\xi_{x,y}$  encodes how much the point violates the margin of the hyperplane.

Joachims' cutting-plane algorithm solves (63) in linear time over the training dataset [20]. The algorithm solves the following *structural classification SVM* formulation, which is equivalent to (63):

$$\begin{aligned} \min_{\beta, \xi \geq 0} \quad & \frac{1}{2} \|\beta\|^2 + C\xi \\ \text{s.t.} \quad & \frac{1}{|G|} \left\langle \beta, \sum_{(x,y) \in T} yx \right\rangle \geq \frac{1}{|G|} |T| - \xi, \quad \forall T \subseteq G. \end{aligned} \quad (64)$$

This formulation has  $2^{|G|}$  constraints, one for each possible subset  $T \subseteq G$ , and a single slack variable  $\xi$  that is shared by all constraints.

Algorithm 1 presents Joachims' cutting-plane algorithm for solving (64). It iteratively constructs a set of constraints  $\mathcal{W}$ , which is a subset of all constraints in (64). In each round  $t$ , it first computes the optimal value for  $\beta^{(t)}$  and  $\xi^{(t)}$  over the current working set  $\mathcal{W}$ . Then, it identifies the constraint  $T^{(t)}$  that is most violated for the current  $\beta^{(t)}$ , and adds this constraint to  $\mathcal{W}$ . It continues until  $T^{(t)}$  is violated by at most  $\epsilon$ . Joachims showed that Algorithm 1 finds the  $\epsilon$ -approximate solution to (64) in  $O(1)$ -many iterations [20]. Hence  $|\mathcal{W}|$  and the number of constraints of the optimization problem are bounded by a number *independent* of  $|G|$ .

Next, we consider the inner optimization problem at line 5. Although  $|\mathcal{W}|$  is small, the number  $n$  of variables can still be large. This prohibits solving with quadratic programming as it can take up to  $O(n^3)$  [27]. Its Wolfe dual, on the other hand, is a quadratic program with only a constant number of variables that is independent of  $n$  and one constraint. Let  $x_T = \sum_{(x,y) \in T} yx$ . We next present the derived Wolfe dual

(its derivation from (64) is in [1]):

$$\begin{aligned} \max_{\alpha \geq 0} \quad & -\frac{1}{2} \left\langle \sum_{T \in \mathcal{W}} \alpha_T x_T, \sum_{T \in \mathcal{W}} \alpha_T x_T \right\rangle + \sum_{T \in \mathcal{W}} |T| \alpha_T \\ \text{s.t.} \quad & \sum_{T \in \mathcal{W}} \alpha_T \leq \frac{C}{|G|} \end{aligned} \quad (65)$$

where  $\alpha = (\alpha_T)_{T \in \mathcal{W}}$  is the vector of constraints.

**Theorem 4.3.** *Let  $I$  be an input database where  $N$  is the largest relation in  $I$ , and  $Q$  be a feature extraction query. A linear SVM classification model can be learned over the training dataset  $Q(I)$  with Joachims' cutting-plane algorithm in time  $\tilde{O}(N^{\#\text{smf}_{w_\ell}(Q)})$  with #PANDA and in time  $O(N^{\text{faq}_{w_\ell}(Q)} \log N)$  with InsideOut.*

#### 4.4 $k$ -means clustering

We next consider  $k$ -means clustering, which is a popular example of an unsupervised machine learning algorithm.

An unsupervised machine learning model is computed over a dataset  $G \subseteq \mathbb{R}^n$ , for which each tuple  $x \in G$  is a vector of features without a label. A clustering task aims to divide  $G$  into  $k$  clusters of "similar" data points with respect to the  $\ell_2$  norm:  $G = \cup_{i=1}^k G_i$ , where  $k$  is a given fixed positive integer. Each cluster  $G_i$  is represented by a cluster mean  $\mu_i \in \mathbb{R}^n$ . One of the most ubiquitous clustering methods, Lloyd's  $k$ -means clustering algorithm (also known as the  $k$ -means method), involves the optimization problem (1) with respect to the partition  $(G_i)_{i \in [k]}$  and the  $k$  means  $(\mu_i)_{i \in [k]}$ . Other norms or distance measures can be used, e.g., if we replace  $\ell_2$  with  $\ell_1$ -norm, then we get the  $k$ -median problem. The subsequent development considers the  $\ell_2$ -norm.

Lloyd's algorithm can be viewed as a special instantiation of the *Expectation-Maximization* (EM) algorithm. It iteratively computes two updating steps until convergence. First, it updates the cluster assignments for each  $(G_i)_{i \in [k]}$ :

$$G_i = \left\{ x \in G \mid \|x - \mu_i\|^2 \leq \|x - \mu_j\|^2, \forall j \in [k] \setminus \{i\} \right\} \quad (66)$$

and then it updates the corresponding  $k$ -means  $(\mu_i)_{i \in [k]}$ :

$$\mu_i = \frac{1}{|G_i|} \sum_{x \in G_i} x. \quad (67)$$

Our observation is that we can reformulate the updating of the  $k$ -means as FAQ-AI expressions, without explicitly computing the partitioning  $(G_i)_{i \in [k]}$ . For a given set of  $k$ -means  $(\mu_j)_{j \in [k]}$ , let  $c_{ij}(x)$  be the following function:

$$\begin{aligned} c_{ij}(x) &= \sum_{\ell \in [n]} [(x_\ell - \mu_{i,\ell})^2 - (x_\ell - \mu_{j,\ell})^2] \\ &= \sum_{\ell \in [n]} [\mu_{i,\ell}^2 - 2x_\ell(\mu_{i,\ell} + \mu_{j,\ell}) - \mu_{j,\ell}^2]. \end{aligned} \quad (68)$$

where  $\mu_{j,\ell}$  is the  $\ell$ 'th component of mean vector  $\mu_j$ . A data point  $\mathbf{x} \in G$  is closest to center  $\mu_i$  if and only if  $c_{ij}(\mathbf{x}) \leq 0$  holds  $\forall j \in [k]$ .

We use this inequality to reformulate the mean vector  $\mu_i$  as  $O(n)$  FAQ-AI expressions. First, we express  $|G_i|$  as:

$$Q_i() = \sum_{\mathbf{x}} \left( \prod_{j \in [k]} \mathbf{1}_{c_{ij}(\mathbf{x}) \leq 0} \right) \left( \prod_{F \in \mathcal{E}_s} R_F(\mathbf{x}_F) \right). \quad (69)$$

Then, for each  $\ell \in [n]$ , the sum  $\sum_{\mathbf{x} \in G_i} x_\ell$  can be reformulated in FAQ-AI as follows (similarly to (4)):

$$Q_{i\ell}() = \sum_{\mathbf{x}} x_\ell \left( \prod_{j \in [k]} \mathbf{1}_{c_{ij}(\mathbf{x}) \leq 0} \right) \left( \prod_{F \in \mathcal{E}_s} R_F(\mathbf{x}_F) \right). \quad (70)$$

Each component  $(\mu_{i,\ell})_{\ell \in [n]}$  equals the division of  $Q_{i\ell}$  by  $Q_i$ .

Overall, the mean vector  $\mu_i$  can be computed with  $O(n)$  FAQ-AI expressions of the form (3).

**Theorem 4.4.** *Let  $I$  be an input database where  $N$  is the largest relation in  $I$ , and  $Q$  be a feature extraction query where  $n$  is the number of its variables. Each iteration of Lloyd's  $k$ -means algorithm can be computed in time  $\tilde{O}(N^{\#\text{smfw}_\ell(Q)})$  with #PANDA and in time  $O(N^{\text{faqw}_\ell(Q)} \log^{k-1} N)$  with InsideOut.*

## ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 682588. LN gratefully acknowledges support from NSF grants CAREER DMS-1351362 and CNS-1409303, Adobe Research and Toyota Research, and a Margaret and Herman Sokol Faculty Award. BM's was supported in part by a Google Research Award, and NSF grants CCF-1830711, CCF-1824303, and CCF-1733873.

## REFERENCES

- [1] Mahmoud Abo Khamis, Ryan R. Curtin, Benjamin Moseley, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2018. On Functional Aggregate Queries with Additive Inequalities. *CoRR* abs/1812.09526 (2018).
- [2] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2018. AC/DC: In-Database Learning Thunderstruck. In *2nd Workshop on Data Mgt for End-To-End ML (DEEM'18)*. 8:1–8:10.
- [3] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2018. In-Database Learning with Sparse Tensors. In *PODS*. 325–340.
- [4] Mahmoud Abo Khamis, Hung Q. Ngo, Dan Olteanu, and Dan Suciu. 2019. Boolean Tensor Decomposition for Conjunctive Queries with Negation. In *ICDT*. 21:1–21:19.
- [5] Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. 2016. FAQ: Questions Asked Frequently. In *PODS*. 13–28.
- [6] Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. 2017. Juggling Functions Inside a Database. *SIGMOD Rec.* 46, 1 (2017), 6–13.
- [7] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2017. What Do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog Have to Do with One Another?. In *PODS*. 429–444.
- [8] Guillaume Bagan, Arnaud Durand, and Etienne Grandjean. 2007. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In *CSL*. 208–222.
- [9] Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. 2011. Orthogonal Range Searching on the RAM, Revisited. In *SoCG*. 1–10.
- [10] Bernard Chazelle. 1988. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.* 17, 3 (1988), 427–462.
- [11] Bernard Chazelle. 1990. Lower bounds for orthogonal range searching. II. The arithmetic model. *J. ACM* 37, 3 (1990), 439–463.
- [12] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. *Computational geometry* (third ed.). Springer-Verlag, Berlin. xii+386 pages. Algorithms and applications.
- [13] Lukasz Golab and M. Tamer Özsu. 2010. *Data Stream Management*. Morgan & Claypool Publishers.
- [14] Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. 2016. Hypertree Decompositions: Questions and Answers. In *PODS*. 57–74.
- [15] Martin Grohe and Dániel Marx. 2014. Constraint Solving via Fractional Edge Covers. *ACM Trans. Alg.* 11, 1 (2014), 4.
- [16] Torsten Grust. 2002. Accelerating XPath Location Steps. In *SIGMOD*. 109–120.
- [17] Terry Halpin and Spencer Rugaber. 2015. *LogiQL: A Query Language for Smart Databases*. CRC Press.
- [18] Joseph M. Hellerstein et al. 2012. The MADlib Analytics Library or MAD Skills, the SQL. *PVLDB* 5, 12 (2012), 1700–1711.
- [19] Anil K. Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters* 31, 8 (2010), 651–666.
- [20] Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *SIGKDD*. 217–226.
- [21] Zuhair Khayyat et al. 2015. Lightning Fast and Space Efficient Inequality Joins. *PVLDB* 8, 13 (2015), 2074–2085.
- [22] Anthony C. Klug. 1988. On conjunctive queries containing inequalities. *J. ACM* 35, 1 (1988), 146–160.
- [23] Paraschos Koutris, Tova Milo, Sudeepa Roy, and Dan Suciu. 2017. Answering Conjunctive Queries with Inequalities. *Theory of Computing Systems* 61, 1 (01 Jul 2017), 2–30.
- [24] Arun Kumar, Jeffrey F. Naughton, and Jignesh M. Patel. 2015. Learning Generalized Linear Models Over Normalized Data. In *SIGMOD*. 1969–1984.
- [25] Nikos Mamoulis. 2011. *Spatial Data Management*. Morgan & Claypool Publishers.
- [26] Dániel Marx. 2013. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM* 60, 6 (2013), Art. 42, 51.
- [27] Kevin P. Murphy. 2013. *Machine learning: a probabilistic perspective*. MIT Press, Cambridge, Mass.
- [28] Hung Q. Ngo. 2018. Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems. In *PODS*. 111–124.
- [29] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2012. Worst-case optimal join algorithms: [extended abstract]. In *PODS*. 37–48.
- [30] Dan Olteanu and Jiewen Huang. 2009. Secondary-storage confidence computation for conjunctive queries with inequalities. In *SIGMOD*. 389–402.
- [31] Luc Segoufin. 2013. Enumerating with Constant Delay the Answers to a Query. In *ICDT*. 10–20.
- [32] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. 2011. Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming* 127, 1 (01 Mar 2011), 3–30.

- [33] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd. 2016. Generalized Low Rank Models. *Foundations and Trends in Machine Learning* 9, 1 (2016), 1–118.
- [34] Todd L. Veldhuizen. 2014. Triejoin: A Simple, Worst-Case Optimal Join Algorithm. In *ICDT*. 96–106.

## A MISSING DETAILS FROM SECTION 3

### A.1 Proof of Theorem 3.11

**Theorem 3.11.** *Any FAQ-AI query  $Q$  of the form (3) on the Boolean semiring can be answered in time  $\tilde{O}(N^{\text{smfw}_\ell(Q)} + |Q|)$ .*

**PROOF.** As in the proof of Theorem 3.5, we first assume there are no free variables; the generalization to  $F \neq \emptyset$  is trivial. When  $F = \emptyset$ , the query (3) is written in Datalog as:

$$Q() \leftarrow \bigwedge_{K \in \mathcal{E}_s} R_K \wedge \bigwedge_{S \in \mathcal{E}_\ell} \left[ \sum_{\nu \in S} \theta_\nu^S \leq 0 \right]. \quad (71)$$

We write  $R_K$  instead of  $R_K(\mathbf{x}_K)$  and  $\theta_\nu^S$  instead of  $\theta_\nu^S(x_\nu)$  to avoid clutter. It will be implicit throughout this proof that the subscript of a factor/function indicates its arguments. To answer query (71), the first step is to find one relation  $S_{\chi(t)}^{(T, \chi)}$  (over variables  $\chi(t)$ ) for every bag  $t \in V(T)$  of every relaxed tree decomposition  $(T, \chi) \in \text{TD}_0^\ell$  such that the relations  $S_{\chi(t)}^{(T, \chi)}$  together form a solution to the following equation:

$$\bigwedge_{K \in \mathcal{E}_s} R_K \equiv \bigvee_{(T, \chi) \in \text{TD}_0^\ell} \bigwedge_{t \in V(T)} S_{\chi(t)}^{(T, \chi)}. \quad (72)$$

Note that the right-hand side of (72) is a *Boolean tensor decomposition* of the left-hand side. The idea of using Boolean tensor decomposition to speed up query evaluation was used in [4] in the context of queries with *disequalities*. Assuming that we can compute the intermediate relations  $S_{\chi(t)}^{(T, \chi)}$  efficiently satisfying (72), then (71) can be answered by answering for each  $(T, \chi) \in \text{TD}_0^\ell$  an intermediate query:

$$Q^{(T, \chi)}() \leftarrow \bigwedge_{t \in V(T)} S_{\chi(t)}^{(T, \chi)} \wedge \bigwedge_{S \in \mathcal{E}_\ell} \left[ \sum_{\nu \in S} \theta_\nu^S \leq 0 \right]. \quad (73)$$

The final answer  $Q$  is obtained by the trivial Datalog rule:

$$Q() \leftarrow \bigvee_{(T, \chi) \in \text{TD}_0^\ell} Q^{(T, \chi)}(). \quad (74)$$

The key point here is that each intermediate query (73) is an FAQ-AI query (3) with  $\text{faqw}_\ell = 1$ , and thus from Theorem 3.5 each one of them can be answered in time  $\tilde{O}(M)$  where

$$M = \max_{(T, \chi) \in \text{TD}_0^\ell} \max_{t \in V(T)} |S_{\chi(t)}^{(T, \chi)}|. \quad (75)$$

It remains to show how to compute tables  $S_{\chi(t)}^{(T, \chi)}$  that form a solution to (72); to do so, we apply distributivity of  $\vee$  over  $\wedge$  to rewrite the right-hand side of (72) as follows. Let  $\mathcal{M}$  be the

collection of *all* maps  $\beta : \text{TD}_0^\ell \rightarrow 2^V$  such that  $\beta(T, \chi) = \chi(t)$  for some  $t \in V(T)$ ; in other words,  $\beta$  selects one bag  $\chi(t)$  out of each tree decomposition  $(T, \chi)$ . Then, from the distributive law we have

$$\bigvee_{(T, \chi) \in \text{TD}_0^\ell} \bigwedge_{t \in V(T)} S_{\chi(t)}^{(T, \chi)} \equiv \bigwedge_{\beta \in \mathcal{M}} \bigvee_{(T, \chi) \in \text{TD}_0^\ell} S_{\beta(T, \chi)}^{(T, \chi)}, \quad (76)$$

which means to solve the relational equation (72) we can instead solve the equation

$$\bigwedge_{\beta \in \mathcal{M}} \bigvee_{(T, \chi) \in \text{TD}_0^\ell} S_{\beta(T, \chi)}^{(T, \chi)} \equiv \bigwedge_{K \in \mathcal{E}_s} R_K. \quad (77)$$

To solve the above equation, for each  $\beta \in \mathcal{M}$  we can find tables  $S_{\beta(T, \chi)}^{(T, \chi)}$  that form a solution to the following equation

$$\bigvee_{(T, \chi) \in \text{TD}_0^\ell} S_{\beta(T, \chi)}^{(T, \chi)} \equiv \bigwedge_{K \in \mathcal{E}_s} R_K. \quad (78)$$

To do that, for each  $\beta \in \mathcal{M}$ , we compute a solution to the following disjunctive Datalog rule:

$$\bigvee_{(T, \chi) \in \text{TD}_0^\ell} W_{\beta(T, \chi)}^{(T, \chi)} \leftarrow \bigwedge_{K \in \mathcal{E}_s} R_K. \quad (79)$$

Once we obtain the relations  $W_{\beta(T, \chi)}^{(T, \chi)}$ , we can semijoin-reduce them against the input relations  $R_K$ , in order to obtain  $S_{\beta(T, \chi)}^{(T, \chi)}$  that solve (77).

Finally, we evaluate each disjunctive Datalog rule (79) by running the PANDA algorithm, which computes the rule in time bounded by  $\tilde{O}(N^{e(\beta)})$ , where

$$e(\beta) = \max_{h \in \text{ED}_{\beta} \cap \Gamma_n} \min_{(T, \chi) \in \text{TD}_0^\ell} h(\beta(T, \chi)). \quad (80)$$

Maximizing over  $\beta \in \mathcal{M}$ , the runtime is bounded by  $\tilde{O}(N^w)$ , where

$$w = \max_{\beta \in \mathcal{M}} e(\beta) \quad (81)$$

$$= \max_{\beta \in \mathcal{M}} \max_{h \in \text{ED}_{\beta} \cap \Gamma_n} \min_{(T, \chi) \in \text{TD}_0^\ell} h(\beta(T, \chi)) \quad (82)$$

$$= \max_{h \in \text{ED}_{\beta} \cap \Gamma_n} \max_{\beta \in \mathcal{M}} \min_{(T, \chi) \in \text{TD}_0^\ell} h(\beta(T, \chi)) \quad (83)$$

$$= \max_{h \in \text{ED}_{\beta} \cap \Gamma_n} \min_{(T, \chi) \in \text{TD}_0^\ell} \max_{t \in V(T)} h(\chi(t)) = \text{smfw}_\ell(Q). \quad (84)$$

The first equality in (84) follows from the minimax lemma in [7]. Our reasoning above also shows that  $M$  from (75) is bounded by  $N^{\text{smfw}_\ell(Q)}$ .  $\square$

### A.2 Proof of Theorem 3.14

**Theorem 3.14.** *Any FAQ query  $Q$  of the form (2) on any semiring can be answered in time  $\tilde{O}(N^{\text{smfw}_\ell(Q)} + |Q|)$ .*

PROOF. The PANDA algorithm [7] takes as input a disjunctive Datalog query of the form

$$\bigvee_{B \in \mathcal{B}} G_B(\mathbf{x}_B) \leftarrow \bigwedge_{K \in \mathcal{E}} R_K(\mathbf{x}_K). \quad (85)$$

The above query has an input relation  $R_K$  for each hyperedge  $K \in \mathcal{E}$  in the query's hypergraph  $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ . The output to the above query is a collection of tables  $G_B$ , one for each “goal” (or “target”)  $B$  in the collection of goals  $\mathcal{B}$ . The output tables  $(G_B)_{B \in \mathcal{B}}$  must satisfy the logical implication in (85): In particular, for each tuple  $\mathbf{x}_{\mathcal{V}}$  that satisfies the conjunction  $\bigwedge_{K \in \mathcal{E}} R_K(\mathbf{x}_K)$ , the disjunction  $\bigvee_{B \in \mathcal{B}} G_B(\mathbf{x}_B)$  must hold. Query (30) is an example of (85). A disjunctive Datalog query (85) can have many valid outputs. The PANDA algorithm computes one such output in time  $\tilde{O}(N^e)$ , where

$$e = \max_{h \in \text{ED}_{\neq} \cap \Gamma_n} \min_{B \in \mathcal{B}} h(B). \quad (86)$$

(Recall notation from Section 2.2.)

In what follows, we describe a variant of PANDA, called #PANDA, that takes a disjunctive Datalog query (85), and computes the following:

- A collection of tables  $(G_B)_{B \in \mathcal{B}}$  that form a valid output to query (85), i.e. that satisfy the logical implication in (85).
- Moreover, associated with each output table  $G_B$ , #PANDA additionally computes a collection of “filter” tables  $(F_K^{(B)})_{K \in \mathcal{E}}$ , one table  $F_K^{(B)}$  for each hyperedge  $K \in \mathcal{E}$  in the input hypergraph  $\mathcal{H}$ . The output tables  $G_B$  along with the associated filters  $(F_K^{(B)})_{K \in \mathcal{E}}$  satisfy the following condition: For each tuple  $\mathbf{x}_{\mathcal{V}}$  that satisfies the conjunction  $\bigwedge_{K \in \mathcal{E}} R_K(\mathbf{x}_K)$ , there is *exactly one* target  $B \in \mathcal{B}$  where the conjunction  $\bigwedge_{K \in \mathcal{E}} F_K^{(B)}(\mathbf{x}_K)$  holds, and for that target  $B$ ,  $G_B(\mathbf{x}_B)$  holds as well. In particular, the following equivalences hold:

$$\bigvee_{B \in \mathcal{B}} \left[ \bigwedge_{K \in \mathcal{E}} F_K^{(B)}(\mathbf{x}_K) \right] \equiv \bigwedge_{K \in \mathcal{E}} R_K(\mathbf{x}_K), \quad (87)$$

$$\left[ \bigwedge_{K \in \mathcal{E}} F_K^{(B)}(\mathbf{x}_K) \right] \equiv \left[ G_B(\mathbf{x}_B) \wedge \bigwedge_{K \in \mathcal{E}} F_K^{(B)}(\mathbf{x}_K) \right], \quad \forall B \in \mathcal{B}, \quad (88)$$

where  $\bigvee$  above denotes the exclusive OR.

#PANDA computes the above output tables  $(G_B)_{B \in \mathcal{B}}$  and  $(F_K^{(B)})_{K \in \mathcal{E}, B \in \mathcal{B}}$  in time  $\tilde{O}(N^{e'})$  where

$$e' = \max_{h \in \text{ED}_{\neq} \cap \Gamma_n} \min_{B \in \mathcal{B}} h(B). \quad (89)$$

Now we briefly explain how to tweak the PANDA algorithm into #PANDA satisfying the above characteristics. We refer the reader to [7] for more details about PANDA. At a

high level, the PANDA algorithm starts with proving an exact upperbound on  $e$  from (86) using a sequence of proof steps, called the *proof sequence*. Then PANDA interprets each step in the proof sequence as a relational operator, and then uses this sequence of relational operators as a query plan to actually compute the query in time  $\tilde{O}(N^e)$ . One of the proof steps used in PANDA is *the decomposition step*  $h(Y) \rightarrow h(X) + h(Y|X)$  for some  $X \subseteq Y \subseteq \mathcal{V}$ . The relational operator corresponding to this decomposition step is the “partitioning” operator, in which we take an input (or intermediate) table  $R_Y$  and partition it into a small number  $k = O(\log |R|)$  of tables  $R_Y^{(1)}, \dots, R_Y^{(k)}$ , based on the degrees of variables in  $Y$  with respect to variables in  $X \subseteq Y$ . In particular, define the degree of  $Y$  w.r.t. a tuple  $\mathbf{t}_X \in \pi_X R_Y$  as

$$\text{deg}_{R_Y}(Y|\mathbf{t}_X) := \left| \left\{ \mathbf{t}'_Y \in R_Y \mid \mathbf{t}'_X = \mathbf{t}_X \right\} \right| \quad (90)$$

In the partitioning step, we partition tuples  $\mathbf{t}_X \in \pi_X R_Y$  into  $k$  buckets based on  $\text{deg}_{R_Y}(Y|\mathbf{t}_X)$  and partition  $R_Y$  accordingly. After partitioning, PANDA creates  $k$  independent branches of the problem, where in the  $j$ -th branch,  $R_Y$  is replaced by  $R_Y^{(j)}$ , for each  $j \in [k]$ . PANDA continues on each branch independently and end up computing a target  $G_B$  for some  $B \in \mathcal{B}$  that is potentially different for each branch.

From the proof sequence construction described in [7], we note the following: If the constructed proof sequence that is used to prove the bound on  $e$  in (86) contains a decomposition step  $h(Y) \rightarrow h(X) + h(Y|X)$ , then the proof of the bound on  $e$  must have relied on some submodularity constraint on  $h$  of the form  $h(X) + h(Z \cup Y) \leq h(Y) + h(Z \cup X)$  for some  $Z \subseteq \mathcal{V}$  where  $Z \cap Y = \emptyset$ . However, the new bound (89) used in #PANDA only relies on submodularities  $h(X) + h(Z \cup Y) \leq h(Y) + h(Z \cup X)$  where  $X \subseteq K$  for some  $K \in \mathcal{E}$ . (Recall  $\Gamma_n|_{\mathcal{E}_{\neq}}$  from Definition 3.12.) Therefore, in #PANDA, whenever we apply a partitioning step of  $R_Y$  into  $R_Y^{(1)}, \dots, R_Y^{(k)}$  based on the degrees  $\text{deg}_{R_Y}(Y|\mathbf{t}_X)$  of  $\mathbf{t}_X \in \pi_X R_Y$ , we can add  $\pi_X R_Y^{(j)}$  into the filter  $F_K^{(B)}$  for some  $K \in \mathcal{E}$ , i.e. we can set  $F_K^{(B)} \leftarrow F_K^{(B)} \bowtie \pi_X R_Y^{(j)}$  on the  $j$ -th branch. Semijoin-reducing  $\pi_X R_Y^{(j)}$  into some  $F_K^{(B)}$  is possible thanks to the fact that  $X \subseteq K$  for some  $K \in \mathcal{E}$ . Moreover, this semijoin-reduction of filters  $F_K^{(B)}$  maintains (87). (Initially, we start with filters  $F_K^{(B)}$  that are identical to the input relations  $R_K$ , which trivially satisfies (87).)

Now that we have described the #PANDA algorithm satisfying the above properties, we explain how to use it as a blackbox to solve an FAQ query  $Q$  of the form (2) in time  $\tilde{O}(N^{\#\text{smfw}(Q)} + |Q|)$ . Following the same notation as in the proof of Theorem 3.11, let  $\mathcal{M}$  be the collection of *all* maps  $\beta : \text{TD}_F^{\mathcal{L}} \rightarrow 2^{\mathcal{V}}$  such that  $\beta(T, \chi) = \chi(t)$  for some  $t \in V(T)$ ; in other words,  $\beta$  selects one bag  $\chi(t)$  out of each tree decomposition  $(T, \chi)$ . For each  $\beta \in \mathcal{M}$ , we use #PANDA to solve



the following rule:

$$\bigvee_{(T, \chi) \in \text{TD}_F} \left[ G_{\beta(T, \chi)} \wedge \bigwedge_{K \in \mathcal{E}} F_K^{(\beta(T, \chi))} \right] \equiv \bigwedge_{K \in \mathcal{E}} R_K. \quad (91)$$

The solutions collectively satisfy the following:

$$\bigwedge_{\beta \in \mathcal{M}} \bigvee_{(T, \chi) \in \text{TD}_F} \left[ G_{\beta(T, \chi)} \wedge \bigwedge_{K \in \mathcal{E}} F_K^{(\beta(T, \chi))} \right] \equiv \bigwedge_{K \in \mathcal{E}} R_K.$$

Now we distribute the outer conjunction  $\bigwedge_{\beta \in \mathcal{M}}$  over the exclusive OR  $\bigvee$ , which results in an exclusive OR outside and a big conjunction inside. Using the same diagonalization argument from [7], we know that for this inner conjunction there must exist some tree decomposition  $(\bar{T}, \bar{\chi}) \in \text{TD}_F$  where the conjunction contains  $G_{\bar{\chi}(t)}$  for all  $t \in V(\bar{T})$ . Thanks to (88), we can keep those terms  $G_{\bar{\chi}(t)}$  in the conjunction and drop out all other terms  $G_{\beta(T, \chi)}$  to get an equivalent conjunction. We interpret the resulting conjunction as an FAQ query: The input factors to this FAQ query are all filter  $F_K^{(\beta(T, \chi))}$  in the conjunction along with  $G_{\bar{\chi}(t)}$  for all  $t \in V(\bar{T})$ ; all other  $G_{\beta(T, \chi)}$  have been dropped. Now we solve this FAQ query by running InsideOut over the tree decomposition  $(\bar{T}, \bar{\chi})$ . We repeat the above for every conjunction. Afterwards, because different conjunctions are joined together with an *exclusive* OR, we can simply add up individual query results.

From (89), the total runtime is  $\tilde{O}(N^w + |Q|)$ , where

$$\begin{aligned} w &= \max_{\beta \in \mathcal{M}} \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n | \mathcal{E}_{\phi_0}} \min_{(T, \chi) \in \text{TD}_F} h(\beta(T, \chi)) \\ &= \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n | \mathcal{E}_{\phi_0}} \max_{\beta \in \mathcal{M}} \min_{(T, \chi) \in \text{TD}_F} h(\beta(T, \chi)) \\ &= \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n | \mathcal{E}_{\phi_0}} \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} h(\chi(t)) = \# \text{smfw}(Q). \end{aligned}$$

□

### A.3 Proof of Proposition 3.15

#### Proposition 3.15.

(a) For any FAQ query  $Q$ , the following holds:

$$\text{smfw}(Q) \leq \# \text{smfw}(Q) \leq \text{faqw}(Q). \quad (92)$$

In particular, when  $Q$  has no free variables, we have

$$\text{subw}(Q) \leq \# \text{subw}(Q) \leq \text{fhtw}(Q). \quad (93)$$

(b) Furthermore, there are classes of queries  $Q$  for which the gap between  $\# \text{smfw}(Q)$  and  $\text{faqw}(Q)$  is unbounded, and so is the gap between  $\# \text{subw}(Q)$  and  $\text{fhtw}(Q)$ .

PROOF. First we prove part (a). The first inequality in (92) follows directly from the definitions of  $\# \text{smfw}$  and  $\text{smfw}$  along with the fact that  $\Gamma_n \subseteq \Gamma_n | \mathcal{E}_{\phi_0}$ . To prove the second inequality in (92), we use the following variant of the *Modularization Lemma* from [7]:

**Claim 1** (Variant of the Modularization Lemma [7]). *Given a hypergraph  $\mathcal{H} = (\mathcal{V} = [n], \mathcal{E})$  and a set  $B \subseteq \mathcal{V}$ , we have*

$$\max_{h \in \text{ED} \cap \Gamma_n | \mathcal{E}} h(B) = \max_{h \in \text{ED} \cap \mathcal{M}_n} h(B), \quad (94)$$

where ED is given by (5) and  $\mathcal{M}_n$  denotes the set of all modular functions  $h : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$ . (A function  $h : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$  is modular if  $h(X) = \sum_{i \in X} h(i)$ ,  $\forall X \subseteq \mathcal{V}$ .)

PROOF OF CLAIM 1. Obviously, the LHS of (94) is lower-bounded by the RHS. Next, we prove LHS  $\leq$  RHS. WLOG we assume  $B = [k]$  for some  $k \in [n]$ . Let  $h^* = \arg \max_{h \in \text{ED} \cap \Gamma_n | \mathcal{E}} h(B)$ .

Define a function  $\bar{h} : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$  as follows:

$$\bar{h}(F) = \sum_{i \in F} (h^*([i]) - h^*([i-1])).$$

Obviously  $\bar{h} \in \mathcal{M}_n$  and  $\bar{h}(B) = h^*(B)$ . Next, we prove  $\bar{h} \in \text{ED}$  by proving that for every  $F \subseteq [n]$  where  $F \subseteq E$  for some  $E \in \mathcal{E}$ , the following holds:  $\bar{h}(F) \leq h^*(F)$ .

The proof is by induction on  $|F|$ . The base case when  $|F| = 0$  is trivial. For the inductive step, consider some  $F$  where  $F \subseteq E$  for some  $E \in \mathcal{E}$ . Let  $j$  be the maximum integer in  $F$ , then by noting that  $|F \cap [j-1]| < |F|$ , we have

$$\begin{aligned} \bar{h}(F) &= h^*([j]) - h^*([j-1]) + \sum_{i \in F - [j]} (h^*([i]) - h^*([i-1])) \\ &= h^*([j]) - h^*([j-1]) + \bar{h}(F \cap [j-1]) \\ &= h^*(F \cup [j-1]) - h^*([j-1]) + \bar{h}(F \cap [j-1]) \\ &\leq h^*(F \cup [j-1]) - h^*([j-1]) + h^*(F \cap [j-1]) \leq h^*(F). \end{aligned}$$

The first inequality above is by induction hypothesis, and the second inequality follows from the fact that  $h^*$  is a  $\mathcal{E}$ -polymatroid (recall Definition 3.12). Both steps rely on the fact that  $F \cap [j-1] \subseteq E$  for some  $E \in \mathcal{E}$ . Consequently,  $\bar{h} \in \text{ED} \cap \mathcal{M}_n$ . Since  $\bar{h}(B) = h^*(B)$ , this proves Claim 1. □

Now we prove the second inequality in (92):

$$\begin{aligned} \# \text{smfw}(Q) &= \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n | \mathcal{E}_{\phi_0}} \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} h(\chi(t)) \\ &\leq \min_{(T, \chi) \in \text{TD}_F} \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n | \mathcal{E}_{\phi_0}} \max_{t \in V(T)} h(\chi(t)) \\ &= \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} \max_{h \in \text{ED}_{\phi_0} \cap \Gamma_n | \mathcal{E}_{\phi_0}} h(\chi(t)) \\ (\text{Claim 1}) &= \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} \max_{h \in \text{ED}_{\phi_0} \cap \mathcal{M}_n} h(\chi(t)) \\ (\text{Strong duality}) &= \min_{(T, \chi) \in \text{TD}_F} \max_{t \in V(T)} \rho_{\mathcal{E}_{\phi_0}}^*(\chi(t)) = \text{faqw}(Q). \end{aligned}$$

The fact that  $\max_{h \in \text{ED}_{\phi_0} \cap \mathcal{M}_n} h(\chi(t)) = \rho_{\mathcal{E}_{\phi_0}}^*(\chi(t))$  follows from the two sides being dual linear programs. (Recall the definition of  $\rho^*$  from Section 2.1.)

Now, we prove part (b) of Proposition A.3. In [7], we constructed a class of graphs/queries where the gap between  $\text{fhtw}$  and  $\text{subw}$  is unbounded. We will re-use the same construction here and prove that the upperbound on  $\text{subw}$  that

we proved in [7] is also an upperbound on #subw. The upperbound proof is going to be different from [7] though since here we can only use  $\mathcal{E}$ -polymatroid properties to prove the bound (recall Definition 3.12).

Given integers  $m$  and  $k$ , consider a graph  $\mathcal{H} = (\mathcal{V}, \mathcal{E})$  which is an “ $m$ -fold  $2k$ -cycle”: The vertex set  $\mathcal{V} := I_1 \cup \dots \cup I_{2k}$  is a disjoint union of  $2k$ -sets of vertices. Each set  $I_j$  has  $m$  vertices in it, i.e.,  $I_j := \{I_j^1, I_j^2, \dots, I_j^m\}$ . There is no edge between any two vertices within the set  $I_j$  for every  $j \in [2k]$ , i.e.,  $I_j$  is an independent set. The edge set  $\mathcal{E}$  of the hypergraph is the union of  $2k$  complete bipartite graphs  $K_{m,m}$ :

$$\mathcal{E} := (I_1 \times I_2) \cup (I_2 \times I_3) \cup \dots \cup (I_{2k-1} \times I_{2k}) \cup (I_{2k} \times I_1).$$

Finally consider an FAQ query  $Q$  that has a finite-sized input factor  $R_K$  for every  $K \in \mathcal{E}$ , i.e.,  $\mathcal{E}_{\varphi_0} = \mathcal{E}$  and  $\mathcal{E}_{\infty} = \emptyset$  (recall notation from Section 2.2). Assuming  $Q$  has no free variables, then  $\text{faqw}(Q) = \text{fhtw}(Q)$  and  $\#\text{smfw}(Q) = \#\text{subw}(Q)$ .

We proved in [7] that  $\text{fhtw}(Q) \geq 2m$ . Next we prove that  $\#\text{subw}(Q) \leq m(2-1/k)$ . Let  $h$  be any function in  $\text{ED}_{\varphi_0} \cap \Gamma_n |_{\mathcal{E}_{\varphi_0}}$ . We recognize two cases:

- Case 1:  $h(I_i) \leq \theta$  for some  $i \in [2k]$ . WLOG assume  $h(I_1) \leq \theta$ . Consider the TD



For bag  $B = I_1 \cup I_i \cup I_{i+1}$ , using  $\mathcal{E}_{\varphi_0}$ -polymatroid properties (Definition 3.12), we have

$$\begin{aligned} h(B) &\leq h(I_1) + h(I_i \cup I_{i+1}) \\ &\leq h(I_1) + \sum_{j=1}^m h(\{I_i^j, I_{i+1}^j\}) \leq \theta + m. \end{aligned}$$

- Case 2:  $h(I_i) > \theta$  for all  $i \in [2k]$ . Consider the TD



Bag  $B_1$

Bag  $B_2$

For convenience, given any vertex  $I_i^j$ , define the vertex set  $\mathcal{V}_i^j$  as follows:

$$\mathcal{V}_i^j := I_1 \cup I_2 \cup \dots \cup I_{i-1} \cup \{I_i^1, I_i^2, \dots, I_i^{j-1}\}.$$

From  $\mathcal{E}_{\varphi_0}$ -polymatroid properties, we have

$$\begin{aligned} h(B_1) &= h(I_1 \cup I_2) + \sum_{i=3}^{k+1} \sum_{j=1}^m h(\{I_i^j\} \cup \mathcal{V}_i^j \mid \mathcal{V}_i^j) \\ &\leq h(I_1 \cup I_2) + \sum_{i=3}^{k+1} \sum_{j=1}^m h(\{I_i^j, I_{i-1}^j\} \mid \{I_{i-1}^j\}) \\ &= h(I_1 \cup I_2) + \sum_{i=3}^{k+1} \sum_{j=1}^m h(\{I_i^j, I_{i-1}^j\}) - \sum_{i=3}^{k+1} \sum_{j=1}^m h(\{I_{i-1}^j\}) \\ &\leq h(I_1 \cup I_2) + \sum_{i=3}^{k+1} \sum_{j=1}^m h(\{I_i^j, I_{i-1}^j\}) - \sum_{i=3}^{k+1} h(I_{i-1}) \end{aligned}$$

$$\leq \sum_{i=2}^{k+1} \sum_{j=1}^m h(\{I_i^j, I_{i-1}^j\}) - \sum_{i=3}^{k+1} h(I_{i-1}) \leq km - (k-1)\theta.$$

In a symmetric way, we can also show that  $h(B_2) \leq km - (k-1)\theta$ . By setting  $\theta = (1-1/k)m$ , we prove that  $\#\text{subw}(Q) \leq m(2-1/k)$ . Since  $\text{fhtw}(Q) \geq 2m$ , this proves part (b).  $\square$

#### A.4 More details on Example 3.19

Consider the count query from Example 3.19. First we prove that  $\#\text{smfw}_{\ell}(Q) \leq 1.5$ . Here  $F = \emptyset$ . We will use two *relaxed* tree decompositions in  $\text{TD}_F^{\ell}$ : The first  $(T_1, \chi_1)$  has two bags  $\{a, b, c\}$  and  $\{c, d\}$ . The second  $(T_2, \chi_2)$  has two bags  $\{a, b\}$  and  $\{b, c, d\}$ . (Both are relaxed TDs because the ligament edge  $1_{a+b+c+d \leq 0}$  is not contained in any bag; recall Definition 3.3.) Following (51), for each  $h \in \text{ED}_{\varphi_0} \cap \Gamma_n |_{\mathcal{E}_{\varphi_0}}$ , we will pick one TD or the other. In particular, given some  $h \in \text{ED}_{\varphi_0} \cap \Gamma_n |_{\mathcal{E}_{\varphi_0}}$ :

- If  $h(b) \geq 1/2$ , then  $h(bc|b) \leq 1/2$ . We pick  $(T_1, \chi_1)$ . From  $\mathcal{E}_{\varphi_0}$ -polymatroid properties (Def. 3.12), we have

$$\begin{aligned} h(abc) &= h(ab) + h(abc|ab) \leq h(ab) + h(bc|b) \leq 1.5, \\ h(cd) &\leq 1. \end{aligned}$$

- If  $h(b) < 1/2$ , we pick  $(T_2, \chi_2)$ .

$$h(ab) \leq 1,$$

$$h(bcd) = h(b) + h(bcd|b) \leq h(b) + h(cd) \leq 1.5.$$

This proves that  $\#\text{smfw}_{\ell}(Q) \leq 1.5$ .

Finally, as a special case of #PANDA, we explain how to solve the above query in time  $\tilde{O}(N^{1.5})$  (where recall  $N := \max\{|R|, |S|, |T|\}$ ). Let

$$S^{\ell} := \{(b, c) \in S \mid |\{c' \mid (b, c') \in S\}| \leq \sqrt{N}\},$$

$$S^h := S \setminus S^{\ell}.$$

Now we can write

$$Q() = \sum_{a,b,c,d} R(a, b) (S^{\ell}(b, c) + S^h(b, c)) T(c, d) \cdot \mathbf{1}_{a+b+c+d \leq 0}$$

$$= Q^{\ell}() + Q^h(), \text{ where}$$

$$Q^{\ell}() := \sum_{a,b,c,d} \underbrace{R(a, b) \cdot S^{\ell}(b, c)}_{U(a,b,c)} \cdot T(c, d) \cdot \mathbf{1}_{a+b+c+d \leq 0},$$

$$Q^h() := \sum_{a,b,c,d} R(a, b) \cdot \underbrace{S^h(b, c) \cdot T(c, d)}_{W(b,c,d)} \cdot \mathbf{1}_{a+b+c+d \leq 0}.$$

Both  $U$  and  $W$  above have sizes  $\leq N^{1.5}$ . Using the algorithm from the proof of Theorem 3.5,  $Q^{\ell}$  can be answered in time  $O(N^{1.5} \log N)$  using the relaxed TD  $(T_1, \chi_1)$ , while  $Q^h$  can be answered in the same time using  $(T_2, \chi_2)$ .