Join Sampling Under Acyclic Degree Constraints and (Cyclic) Subgraph Sampling

Ru Wang
The Chinese University of Hong Kong, China

Yufei Tao
The Chinese University of Hong Kong, China

Abstract

Given a (natural) join with an acyclic set of degree constraints (the join itself does not need to be acyclic), we show how to draw a uniformly random sample from the join result in $O(polymat/\max\{1, \text{OUT}\})$ expected time (assuming data complexity) after a preprocessing phase of $O(\text{IN})$ expected time, where IN, OUT, and *polymat* are the join's input size, output size, and polymatroid bound, respectively. This compares favorably with the state of the art (Deng et al. and Kim et al., both in PODS'23), which states that, in the absence of degree constraints, a uniformly random sample can be drawn in $\tilde{O}(AGM/\max\{1, \text{OUT}\})$ expected time after a preprocessing phase of $\tilde{O}(\text{IN})$ expected time, where AGM is the join's AGM bound and $\tilde{O}(.)$ hides a polylog(IN) factor. Our algorithm applies to every join supported by the solutions of Deng et al. and Kim et al. Furthermore, since the polymatroid bound is at most the AGM bound, our performance guarantees are never worse, but can be considerably better, than those of Deng et al. and Kim et al.

We then utilize our techniques to tackle *directed subgraph sampling*, a problem that has extensive database applications and bears close relevance to joins. Let $G = (V, E)$ be a directed data graph where each vertex has an out-degree at most λ, and let P be a directed pattern graph with a constant number of vertices. The objective is to uniformly sample an occurrence of P in G. The problem can be modeled as join sampling with input size $\text{IN} = \Theta(|E|)$ but, whenever P contains cycles, the converted join has *cyclic* degree constraints. We show that it is always possible to throw away certain degree constraints such that (i) the remaining constraints are acyclic and (ii) the new join has asymptotically the same polymatroid bound *polymat* as the old one. Combining this finding with our new join sampling solution yields an algorithm to sample from the original (cyclic) join (thereby yielding a uniformly random occurrence of P) in $O(polymat/\max\{1, \text{OUT}\})$ expected time after $O(|E|)$ expected-time preprocessing, where OUT is the number of occurrences.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Information systems → Join algorithms

Keywords and phrases Join Sampling, Subgraph Sampling, Degree Constraints, Polymatroid Bounds

Digital Object Identifier 10.4230/LIPIcs.ICDT.2024.23

Related Version *Full Version*: `https://arxiv.org/abs/2312.12797`

Funding This work was supported in part by GRF projects 14207820, 14203421, and 14222822 from HKRGC.

1 Introduction

In relational database systems, (natural) joins are acknowledged as notably computation-intensive, with its cost surging drastically in response to expanding data volumes. In the current big data era, the imperative to circumvent excessive computation increasingly overshadows the requirement for complete join results. A myriad of applications, including machine learning algorithms, online analytical processing, and recommendation systems, can operate effectively with random samples. This situation has sparked research initiatives focused on devising techniques capable of producing samples from a join result significantly

faster than executing the join in its entirety. In the realm of graph theory, the significance of join operations is mirrored in their intrinsic connections to *subgraph listing*, a classical problem that seeks to pinpoint all the occurrences of a pattern P (for instance, a directed 3-vertex cycle) within a data graph G (such as a social network where a directed edge symbolizes a "follow" relationship). Analogous to joins, subgraph listing demands a vast amount of computation time, which escalates rapidly with the sizes of G and P. Fortunately, many social network analyses do not require the full set of occurrences of P, but can function well with only samples from those occurrences. This has triggered the development of methods that can extract samples considerably faster than finding all the occurrences.

This paper will revisit *join sampling* and *subgraph sampling* under a unified "degree-constrained framework". Next, we will first describe the framework formally in Section 1.1, review the previous results in Section 1.2, and then overview our results in Section 1.3.

1.1 Problem Definitions

Join Sampling. Let **att** be a finite set, with each element called an *attribute*, and **dom** be a countably infinite set, with each element called a *value*. For a non-empty set $\mathcal{X} \subseteq \mathbf{att}$ of attributes, a *tuple* over X is a function $\boldsymbol{u} : \mathcal{X} \to \mathbf{dom}$. For any non-empty subset $\mathcal{Y} \subseteq \mathcal{X}$, we define $\boldsymbol{u}[\mathcal{Y}]$ – the *projection* of $\boldsymbol{u}$ on Y – as the tuple $\boldsymbol{v}$ over $\mathcal{Y}$ satisfying $\boldsymbol{v}(Y) = \boldsymbol{u}(Y)$ for every attribute $Y \in \mathcal{Y}$.

A *relation* R is a set of tuples over the same set $\mathcal{Z}$ of attributes; we refer to $\mathcal{Z}$ as the *schema* of R and represent it as $schema(R)$. The *arity* of R is the size of $schema(R)$. For any subsets $\mathcal{X}$ and $\mathcal{Y}$ of $schema(R)$ satisfying $\mathcal{X} \subset \mathcal{Y}$ (note: $\mathcal{X}$ is a *proper* subset of $\mathcal{Y}$), define:

$$deg_{\mathcal{Y}|\mathcal{X}}(R) \quad = \quad \max_{\text{tuple } \boldsymbol{u} \text{ over } \mathcal{X}} \left| \left\{ \boldsymbol{v}[\mathcal{Y}] \mid \boldsymbol{v} \in R, \boldsymbol{v}[\mathcal{X}] = \boldsymbol{u} \right\} \right|. \tag{1}$$

For an intuitive explanation, imagine grouping the tuples of R by $\mathcal{X}$ and counting, for each group, how many *distinct* $\mathcal{Y}$-projections are formed by the tuples therein. Then, the value $deg_{\mathcal{Y}|\mathcal{X}}(R)$ corresponds to the maximum count of all groups. It is worth pointing out that, when $\mathcal{X} = \emptyset$, then $deg_{\mathcal{Y}|\mathcal{X}}(R)$ is simply $|\Pi_{\mathcal{Y}}(R)|$ where Π is the standard "projection" operator in relational algebra. If in addition $\mathcal{Y} = schema(R)$, then $deg_{\mathcal{Y}|\mathcal{X}}(R)$ equals $|R|$.

We define a *join* as a set $\mathcal{Q}$ of relations (some of which may have the same schema). Let $schema(\mathcal{Q})$ be the union of the attributes of the relations in $\mathcal{Q}$, i.e., $schema(\mathcal{Q}) = \bigcup_{R \in Q} schema(R)$. Focusing on "data complexity", we consider only joins where both $\mathcal{Q}$ and $schema(\mathcal{Q})$ have constant sizes. The result of $\mathcal{Q}$ is a relation over $schema(\mathcal{Q})$ formalized as:

$$join(\mathcal{Q}) = \{\text{tuple } \boldsymbol{u} \text{ over } schema(\mathcal{Q}) \mid \forall R \in \mathcal{Q} : \boldsymbol{u}[schema(R)] \in R\}.$$

Define $\text{IN} = \sum_{R \in \mathcal{Q}} |R|$ and $\text{OUT} = |join(\mathcal{Q})|$. We will refer to IN and OUT as the *input size* and *output size* of Q, respectively.

A *join sampling* operation returns a tuple drawn uniformly at random from $join(\mathcal{Q})$ or declares $join(\mathcal{Q}) = \emptyset$. All such operations must be mutually independent. The objective of the *join sampling problem* is to preprocess the input relations of $\mathcal{Q}$ into an appropriate data structure that can be used to perform join-sampling operations repeatedly.

We study the problem in the scenario where $\mathcal{Q}$ conforms to a set DC of degree constraints. Specifically, each *degree constraint* has the form $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ where $\mathcal{X}$ and $\mathcal{Y}$ are subsets of $schema(\mathcal{Q})$ satisfying $\mathcal{X} \subset \mathcal{Y}$ and $N_{\mathcal{Y}|\mathcal{X}} \geq 1$ is an integer. A relation $R \in \mathcal{Q}$ is said to *guard* the constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ if

$$\mathcal{Y} \subseteq schema(R), \text{ and } deg_{\mathcal{Y}|\mathcal{X}}(R) \leq N_{\mathcal{Y}|\mathcal{X}}.$$

The join $\mathcal{Q}$ is *consistent* with DC – written as $\mathcal{Q} \models \mathsf{DC}$ – if every degree constraint in DC is guarded by at least one relation in $\mathcal{Q}$. It is safe to assume that DC does not have two constraints $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ and $(\mathcal{X}', \mathcal{Y}', N_{\mathcal{Y}'|\mathcal{X}'})$ with $\mathcal{X} = \mathcal{X}'$ and $\mathcal{Y} = \mathcal{Y}'$; otherwise, assuming $N_{\mathcal{Y}|\mathcal{X}} \leq N_{\mathcal{Y}'|\mathcal{X}'}$, the constraint $(\mathcal{X}', \mathcal{Y}', N_{\mathcal{Y}'|\mathcal{X}'})$ is redundant and can be removed from DC.

In this work, we concentrate on "acyclic" degree dependency. To formalize this notion, let us define a *constraint dependency graph* G_{DC} as follows. This is a directed graph whose vertex set is $\mathit{schema}(\mathcal{Q})$ (i.e., each vertex of G_{DC} is an attribute in $\mathit{schema}(\mathcal{Q})$). For each degree constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ such that $\mathcal{X} \neq \emptyset$, we add a (directed) edge (X, Y) to G_{DC} for every pair $(X, Y) \in \mathcal{X} \times (\mathcal{Y} - \mathcal{X})$. We say that the set DC is *acyclic* if G_{DC} is an acyclic graph; otherwise, DC is *cyclic*.

It is important to note that each relation $R \in \mathcal{Q}$ implicitly defines a special degree constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ where $\mathcal{X} = \emptyset$, $\mathcal{Y} = \mathit{schema}(R)$, and $N_{\mathcal{Y}|\mathcal{X}} = |R|$. Such a constraint – known as a *cardinality constraint* – is always assumed to be present in DC. As all cardinality constraints have $\mathcal{X} = \emptyset$, they do not affect the construction of G_{DC}. Consequently, if DC only contains cardinality constraints, then G_{DC} is empty and hence trivially acyclic. Moreover, readers should avoid the misconception that "an acyclic G_{DC} implies $\mathcal{Q}$ being an acyclic join"; these two acyclicity notions are unrelated. While the definition of an acyclic join is not needed for our discussion, readers unfamiliar with this term may refer to [2, Chapter 6.4].

Directed Graph Sampling. We are given a *data graph* $G = (V, E)$ and a *pattern graph* $P = (V_P, E_P)$, both being simple directed graphs. The pattern graph is weakly-connected[1] and has a constant number of vertices. A simple directed graph $G_{sub} = (V_{sub}, E_{sub})$ is a *subgraph* of G if $V_{sub} \subseteq V$ and $E_{sub} \subseteq E$. The subgraph G_{sub} is an *occurrence* of P if they are isomorphic, namely, there is a bijection $f : V_{sub} \to V_P$ such that, for any distinct vertices $u_1, u_2 \in V_{sub}$, there is an edge $(u_1, u_2) \in E_{sub}$ if and only if $(f(u_1), f(u_2))$ is an edge in E_P. We will refer to f as a *isomorphism bijection* between P and G_{sub}.

A *subgraph sampling* operation returns an occurrence of P in G uniformly at random or declares the absence of any occurrence. All such operations need to be mutually independent. The objective of the *subgraph sampling problem* is to preprocess G into a data structure that can support every subgraph-sampling operation efficiently. We will study the problem under a degree constraint: every vertex in G has an out-degree at most λ.

Math Conventions. For an integer $x \geq 1$, the notation $[x]$ denotes the set $\{1, 2, ..., x\}$; as a special case, $[0]$ represents the empty set. Every logarithm $\log(\cdot)$ has base 2, and function $\exp_2(x)$ is defined to be 2^x. We use double curly braces to represent multi-sets, e.g., $\{\!\{1, 1, 1, 2, 2, 3\}\!\}$ is a multi-set with 6 elements.

1.2 Related Work

Join Computation. Any algorithm correctly answering a join query $\mathcal{Q}$ must incur $\Omega(\text{OUT})$ time just to output the OUT tuples in $\mathit{join}(\mathcal{Q})$. Hence, finding the greatest possible value of OUT is an imperative step towards unraveling the time complexity of join evaluation. A classical result in this regard is the *AGM bound* [6]. To describe this bound, let us define the *schema graph* of $\mathcal{Q}$ as a multi-hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where

$$\mathcal{V} = \mathit{schema}(\mathcal{Q}), \text{ and } \mathcal{E} = \{\!\{\mathit{schema}(R) \mid R \in \mathcal{Q}\}\!\}. \tag{2}$$

[1] Namely, if we ignore the edge directions, then P becomes a connected undirected graph.

Note that $\mathcal{E}$ is a multi-set because the relations in $\mathcal{Q}$ may have identical schemas. A *fractional edge cover* of $\mathcal{G}$ is a function $w : \mathcal{E} \to [0, 1]$ such that, for any $X \in \mathcal{V}$, it holds that $\sum_{F \in \mathcal{E}: X \in F} w(F) \geq 1$ (namely, the total weight assigned to the hyperedges covering X is at least 1). Atserias, Grohe, and Marx [6] showed that, given any fractional edge cover, it always holds that $\text{OUT} \leq \prod_{F \in \mathcal{E}} |R_F|^{w(F)}$, where R_F is the relation in $\mathcal{Q}$ whose schema corresponds to the hyperedge F. The AGM bound is defined as $AGM(\mathcal{Q}) = \min_w \prod_{F \in \mathcal{E}} |R_F|^{w(F)}$.

The AGM bound is tight: given any hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and any set of positive integers $\{N_F \mid F \in \mathcal{E}\}$, there is always a join $\mathcal{Q}$ such that $\mathcal{Q}$ has $\mathcal{G}$ as the schema graph, $|R_F| = |N_F|$ for each $F \in \mathcal{E}$, and the output size OUT is $\Theta(AGM(\mathcal{Q}))$. This has motivated the development of algorithms [5, 13, 20, 22, 24, 27, 30–33, 36] that can compute $join(\mathcal{Q})$ in $\tilde{O}(AGM(\mathcal{Q}))$ time – where $\tilde{O}(.)$ hides a factor polylogarithmic to the input size IN of $\mathcal{Q}$ – and therefore are worst-case optimal up to an $\tilde{O}(1)$ factor.

However, the tightness of the AGM bound relies on the assumption that all the degree constraints on $\mathcal{Q}$ are purely cardinality constraints. In reality, general degree constraints are prevalent, and their inclusion could dramatically decrease the maximum output size OUT. This observation has sparked significant interest [12, 16, 20, 21, 23, 24, 29, 34] in establishing refined upper bounds on OUT tailored for more complex degree constraints. Most notably, Khamis et al. [24] proposed the *entropic bound*, which is applicable to any set DC of degree constraints and is tight in a strong sense (see Theorem 5.5 of [34]). Unfortunately, the entropic bound is difficult to compute because it requires solving a linear program (LP) involving infinitely many constraints (it remains an open problem whether the computation is decidable). Not coincidentally, no join algorithm is known to have a running time matching the entropic bound.

To circumvent the above issue, Khamis et al. [24] introduced the *polymatroid bound* as an alternative, which we represent as $polymat(\mathsf{DC})$ because this bound is fully decided by DC (i.e., any join $\mathcal{Q} \models \mathsf{DC}$ must satisfy $\text{OUT} \leq polymat(\mathsf{DC})$). Section 2 will discuss $polymat(\mathsf{DC})$ in detail; for now, it suffices to understand that (i) the polymatroid bound, although possibly looser than the entropic bound, never exceeds the AGM bound, and (ii) $polymat(\mathsf{DC})$ can be computed in $O(1)$ time under data complexity. Khamis et al. [24] proposed an algorithm named PANDA that can evaluate an arbitrary join $\mathcal{Q} \models \mathsf{DC}$ in time $\tilde{O}(polymat(\mathsf{DC}))$.

Interestingly, when DC is acyclic, the entropic bound is equivalent to the polymatroid bound [29]. In this scenario, Ngo [29] presented a simple algorithm to compute any join $\mathcal{Q} \models \mathsf{DC}$ in $O(polymat(\mathsf{DC}))$ time, after a preprocessing of $O(\text{IN})$ expected time.

Join Sampling. For an acyclic join (not to be confused with a join having an acyclic set of degree constraints), it is possible to sample from the join result in constant time, after a preprocessing of $O(\text{IN})$ expected time [38]. The problem becomes more complex when dealing with an arbitrary (cyclic) join $\mathcal{Q}$, with the latest advancements presented in two PODS'23 papers [13, 25]. Specifically, Kim et al. [25] described how to sample in $\tilde{O}(AGM(\mathcal{Q})/\max\{1, \text{OUT}\})$ expected time, after a preprocessing of $\tilde{O}(\text{IN})$ time. Deng et al. [13] achieved the same guarantees using different approaches, and offered a rationale explaining why the expected sample time $O(AGM(\mathcal{Q})/\text{OUT})$ can no longer be significantly improved, even when $0 < \text{OUT} \ll AGM(\mathcal{Q})$, subject to commonly accepted conjectures. We refer readers to [3, 9, 10, 13, 25, 38] and the references therein for other results (now superseded) on join sampling.

Subgraph Listing. Let us start by clarifying the *fractional edge cover number* $\rho^*(P)$ of a simple undirected pattern graph $P = (V_P, E_P)$. Given a fractional edge cover of P (i.e., function $w : E_P \to [0, 1]$ such that, for any vertex $X \in V_P$, we have $\sum_{F \in E_P: X \in F} w(F) \geq 1$),

define $\sum_{F \in E_P} w(F)$ as the *total weight* of w. The value of $\rho^*(P)$ is the smallest total weight of all fractional edge covers of P. Given a directed pattern graph P, we define its fractional edge cover number $\rho^*(P)$ as the value $\rho^*(P')$ of the corresponding undirected graph P' that is obtained from P by ignoring all the edge directions.

When P has a constant size, it is well-known [4, 6] that any data graph $G = (V, E)$ can encompass $O(|E|^{\rho^*(P)})$ occurrences of P. This upper bound is tight: for any integer m, there is a data graph $G = (V, E)$ with $|E| = m$ edges that has $\Omega(m^{\rho^*(P)})$ occurrences of P. Thus, a subgraph listing algorithm is considered worst-case optimal if it finishes in $\tilde{O}(|E|^{\rho^*(P)})$ time.

It is well-known that subgraph listing can be converted to a join $\mathcal{Q}$ on binary relations (namely, relations of arity 2). The join $\mathcal{Q}$ has an input size of $\text{IN} = \Theta(|E|)$, and its AGM bound is $AGM(\mathcal{Q}) = \Theta(|E|^{\rho^*(P)})$. All occurrences of P in G can be derived from $join(\mathcal{Q})$ for free. Thus, any $\tilde{O}(AGM(\mathcal{Q}))$-time join algorithm is essentially worst-case optimal for subgraph listing.

Assuming P and G to be directed, Jayaraman et al. [18] presented interesting enhancement over the above transformation in the scenario where each vertex of G has an out-degree at most λ. The key lies in examining the polymatroid bound of the join $\mathcal{Q}$ derived from subgraph listing. As will be explained in Section 4, this join $\mathcal{Q}$ has a set DC of degree constraints whose constraint dependency graph G_{DC} coincides with P. Jayaraman et al. developed an algorithm that lists all occurrences of $\mathcal{Q}$ in G in $O(polymat(\mathsf{DC}))$ time (after a preprocessing of $O(\text{IN})$ expected time) and confirmed that this is worst-case optimal. Their findings are closely related to our work, and we will delve into them further when their specifics become crucial to our discussion.

There is a substantial body of literature on bounding the cost of subgraph listing using parameters distinct from those already mentioned. These studies typically concentrate on specific patterns (such as paths, cycles, and cliques) or particular graphs (for instance, those that are sparse under a suitable metric). We refer interested readers to [1, 7, 8, 11, 14, 17, 19, 26, 28, 35] and the references therein.

Subgraph Sampling. Fichtenberger, Gao, and Peng [15] described how to sample an occurrence of the pattern P in the data graph G in $O(|E|^{\rho^*(P)}/\max\{1, \text{OUT}\})$ expected time, where OUT is the number of occurrences of P in G, after a preprocessing of $O(|E|)$ expected time. In [13], Deng et al. clarified how to deploy an arbitrary join sampling algorithm to perform subgraph sampling; their approach ensures the same guarantees as in [15], baring an $\tilde{O}(1)$ factor.

1.3 Our Results

For any join $\mathcal{Q}$ with an acyclic set DC of degree constraints, we will demonstrate in Section 3 how to extract a uniformly random sample from $join(\mathcal{Q})$ in $O(polymat(\mathsf{DC})/\max\{1, \text{OUT}\})$ expected time, following an initial preprocessing of $O(\text{IN})$ expected time. This performance is favorable when compared to the recent results of [13, 25] (reviewed in Section 1.2), which examined settings where DC consists only of cardinality constraints and is therefore trivially acyclic. As $polymat(\mathsf{DC})$ is at most but can be substantially lower than $AGM(\mathcal{Q})$, our guarantees are never worse, but can be considerably better, than those in [13, 25].

What if DC is cyclic? An idea, proposed in [29], is to discard enough constraints to make the remaining set DC' of constraints acyclic (while ensuring $\mathcal{Q} \models \mathsf{DC}'$). Our algorithm can then be applied to draw a sample in $O(polymat(\mathsf{DC}')/\max\{1, \text{OUT}\})$ time. However, this can be unsatisfactory because $polymat(\mathsf{DC}')$ can potentially be much larger than $polymat(\mathsf{DC})$.

Our next contribution is to prove that, interestingly, the issue does not affect subgraph listing/sampling. Consider first directed subgraph listing, defined by a pattern graph P and a data graph G where every vertex has an out-degree at most λ. This problem can be converted to a join $\mathcal{Q}$ on binary relations, which is associated with a set DC of degree constraints such that the constraint dependency graph G_{DC} is exactly P. Consequently, whenever P contains a cycle, so does G_{DC}, making DC cyclic. Nevertheless, we will demonstrate in Section 4 the existence of an acyclic set $\mathsf{DC}' \subset \mathsf{DC}$ ensuring $Q \models \mathsf{DC}'$ and $\mathit{polymat}(\mathsf{DC}) = \Theta(\mathit{polymat}(\mathsf{DC}'))$. This "magical" DC' has an immediate implication: Ngo's join algorithm in [29], when applied to Q and DC' directly, already solves directed subgraph listing optimally in $O(\mathit{polymat}(\mathsf{DC}'))$ $= O(\mathit{polymat}(\mathsf{DC}))$ time. This dramatically simplifies – in terms of both procedure and analysis – an algorithm of Jayaraman et al. [18] (for directed subgraph listing, reviewed in Section 1.2) that has the same guarantees.

The same elegance extends to directed subgraph sampling: by applying our new join sampling algorithm to $\mathcal{Q}$ and the "magical" DC', we can sample an occurrence of P in G using $O(\mathit{polymat}(\mathsf{DC})/\max\{1, \mathrm{OUT}\})$ expected time, after a preprocessing of $O(|E|)$ expected time. As $\mathit{polymat}(\mathsf{DC})$ never exceeds but can be much lower than $AGM(\mathcal{Q}) = \Theta(|E|^{\rho^*(P)})$, our result compares favorably with the state of the art [13, 15, 25] reviewed in Section 1.2.

In the full version of this paper [37], we will prove similar results for *undirected subgraph sampling* and demonstrate how our techniques can be significantly simplified in that scenario. By virtue of the power of sampling, our findings have further implications on other fundamental problems including output-size estimation, output permutation, and small-delay enumeration, as discussed in Section 5.

2 Preliminaries

Set Functions, Polymatroid Bounds, and Modular Bounds. Suppose that $\mathcal{S}$ is a finite set. We refer to a function $h : 2^{\mathcal{S}} \to \mathbb{R}_{\geq 0}$ as a *set function over* $\mathcal{S}$, where $\mathbb{R}_{\geq 0}$ is the set of non-negative real values. Such a function h is said to be

- *zero-grounded* if $h(\emptyset) = 0$;
- *monotone* if $h(\mathcal{X}) \leq h(\mathcal{Y})$ for all $\mathcal{X}, \mathcal{Y}$ satisfying $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{S}$;
- *modular* if $h(\mathcal{X}) = \sum_{A \in \mathcal{X}} h(\{A\})$ holds for any $\mathcal{X} \subseteq \mathcal{S}$;
- *submodular* if $h(\mathcal{X} \cup \mathcal{Y}) + h(\mathcal{X} \cap \mathcal{Y}) \leq h(\mathcal{X}) + h(\mathcal{Y})$ holds for any $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{S}$.

Define:

$$\mathsf{M}_{\mathcal{S}} = \text{the set of modular set functions over } \mathcal{S}$$
$$\Gamma_{\mathcal{S}} = \text{the set of set functions over } \mathcal{S} \text{ that are zero-grounded, monotone, submodular}$$

Note that every modular function must be zero-grounded and monotone. Clearly, $\mathsf{M}_{\mathcal{S}} \subseteq \Gamma_{\mathcal{S}}$.

Consider $\mathcal{C}$ to be a set of triples, each having the form $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ where $X \subset \mathcal{Y} \subseteq \mathcal{S}$ and $N_{\mathcal{Y}|\mathcal{X}} \geq 1$ is an integer. We will refer to $\mathcal{C}$ as a *rule collection* over $\mathcal{S}$ and to each triple therein as a *rule*. Intuitively, the presence of a rule collection is to instruct us to focus only on certain restricted set functions. Formally, these are the set functions in:

$$\mathcal{H}_{\mathcal{C}} = \big\{\text{set function } h \text{ over } \mathcal{S} \mid h(\mathcal{Y}) - h(\mathcal{X}) \leq \log N_{\mathcal{Y}|\mathcal{X}}, \ \ \forall (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathcal{C}\big\}. \tag{3}$$

The *polymatroid bound* of $\mathcal{C}$ can now be defined as

$$\mathit{polymat}(\mathcal{C}) = \exp_2\Big(\max_{h \in \Gamma_{\mathcal{S}} \cap \mathcal{H}_{\mathcal{C}}} h(\mathcal{S})\Big). \tag{4}$$

Recall that $\exp_2(x) = 2^x$. Similarly, the *modular bound* of $\mathcal{C}$ is defined as

$$\mathit{modular}(\mathcal{C}) = \exp_2\Big(\max_{h \in \mathsf{M}_{\mathcal{S}} \cap \mathcal{H}_{\mathcal{C}}} h(\mathcal{S})\Big). \tag{5}$$

Join Output Size Bounds. Let us fix a join $\mathcal{Q}$ whose schema graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Suppose that $\mathcal{Q}$ is consistent with a set DC of degree constraints, i.e., $\mathcal{Q} \models \mathsf{DC}$. As explained in Section 1.1, we follow the convention that each relation of $\mathcal{Q}$ implicitly inserts a cardinality constraint (i.e., a special degree constraint) to DC. Note that the set DC is merely a rule collection over $\mathcal{V}$. The following lemma was established by Khamis et al. [24]:

▶ **Lemma 1** ([24]). *The output size* OUT *of* $\mathcal{Q}$ *is at most polymat*(DC)*, i.e., the polymatroid bound of* DC *(as defined in* (4)*).*

How about *modular*(DC), i.e., the modular bound of $\mathcal{V}$? As $\mathsf{M}_\mathcal{V} \subseteq \Gamma_\mathcal{V}$, we have *modular*(DC) $\leq$ *polymat*(DC) and the inequality can be strict in general. However, an exception arises when DC is acyclic, as proved in [29]:

▶ **Lemma 2** ([29]). *When* DC *is acyclic, it always holds that modular*(DC) = *polymat*(DC)*, namely,* $\max_{h \in \Gamma_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V}) = \max_{h \in \mathsf{M}_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$.

As a corollary, when DC is acyclic, the value of *modular*(DC) always serves as an upper bound of OUT. In our technical development, we will need to analyze the set functions $h^* \in \Gamma_\mathcal{V}$ that realize the polymatriod bound, i.e., $h^*(\mathcal{V}) = \max_{h \in \Gamma_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$. A crucial advantage provided by Lemma 2 is that we can instead scrutinize those set functions $h^* \in \mathsf{M}_\mathcal{V}$ realizing the *modular* bound, i.e., $h^*(\mathcal{V}) = \max_{h \in \mathsf{M}_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$. Compared to their submodular counterparts, modular set functions exhibit more regularity because every $h \in \mathsf{M}_\mathcal{V}$ is fully determined by its value $h(\{A\})$ on each *individual* attribute $A \in \mathcal{V}$. In particular, for any $h \in \mathsf{M}_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}$, it holds true that $h(\mathcal{Y}) - h(\mathcal{X}) = \sum_{A \in \mathcal{Y} - \mathcal{X}} h(A)$ for any $\mathcal{X} \subset \mathcal{Y} \subseteq \mathcal{V}$. If we associate each $A \in \mathcal{V}$ with a variable ν_A, then $\max_{h \in \mathsf{M}_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$ – hence, also $\max_{h \in \Gamma_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$ – is precisely the optimal value of the following LP:

$$\textbf{modular LP} \quad \max \sum_{A \in \mathcal{V}} \nu_A \text{ subject to}$$
$$\sum_{A \in \mathcal{Y} - \mathcal{X}} \nu_A \leq \log N_{\mathcal{Y}|\mathcal{X}} \qquad \forall (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$$
$$\nu_A \geq 0 \qquad \forall A \in \mathcal{V}$$

We will also need to work with the LP's dual. Specifically, if we associate a variable $\delta_{\mathcal{Y}|\mathcal{X}}$ for every degree constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$, then the dual LP is:

$$\textbf{dual modular LP} \quad \min \sum_{(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} \delta_{\mathcal{Y}|\mathcal{X}} \cdot \log N_{\mathcal{Y}|\mathcal{X}} \text{ subject to}$$
$$\sum_{\substack{(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC} \\ A \in \mathcal{Y} - \mathcal{X}}} \delta_{\mathcal{Y}|\mathcal{X}} \geq 1 \qquad \forall A \in \mathcal{V}$$
$$\delta_{\mathcal{Y}|\mathcal{X}} \geq 0 \qquad \forall (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$$

3 Join Sampling under Acyclic Degree Dependency

This section serves as a proof of our first main result:

▶ **Theorem 3.** *For any join* $\mathcal{Q}$ *consistent with an acyclic set* DC *of degree constraints, we can build in* $O(\text{IN})$ *expected time a data structure that supports each join sampling operation in* $O(\textit{polymat}(\mathsf{DC})/\max\{1, \text{OUT}\})$ *expected time, where* IN *and* OUT *are the input and out sizes of* $\mathcal{Q}$*, respectively, and polymat*(DC) *is the polymatroid bound of* DC*.*

Basic Definitions. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the schema graph of $\mathcal{Q}$, and G_{DC} be the constraint dependency graph determined by DC. For each hyperedge $F \in \mathcal{E}$, we denote by R_F the relation whose schema corresponds to F. Recall that every constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$ is guarded by at least one relation in $\mathcal{Q}$. Among them, we arbitrarily designate one relation as the constraint's *main guard*, whose schema is represented as $F(\mathcal{X}, \mathcal{Y})$ (the main guard can then be conveniently identified as $R_{F(\mathcal{X},\mathcal{Y})}$).

Set $k = |\mathcal{V}|$. As G_{DC} is a DAG (acyclic directed graph), we can order its k vertices (i.e., attributes) into a topological order: $A_1, A_2, ..., A_k$. For each $i \in [k]$, define $\mathcal{V}_i = \{A_1, A_2, ..., A_i\}$; specially, define $\mathcal{V}_0 = \emptyset$. For any $i \in [k]$, define

$$\mathsf{DC}(A_i) \quad = \quad \{(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC} \mid A_i \in \mathcal{Y} - \mathcal{X}\} \tag{6}$$

Fix an arbitrary $i \in [k]$ and an arbitrary constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)$. Given a tuple $\boldsymbol{w}$ over $\mathcal{V}_{i-1}$ (note: if $i = 1$, then $\mathcal{V}_{i-1} = \emptyset$ and $\boldsymbol{w}$ is a null tuple) and a value $a \in \mathbf{dom}$, we define a "relative degree" for a as:

$$\mathit{reldeg}_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a) \quad = \quad \frac{\left|\sigma_{A_i=a}(\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}))\right|}{\left|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})\right|} \tag{7}$$

where σ and $\ltimes$ are the standard selection and semi-join operators in relational algebra, respectively. To understand the intuition behind $\mathit{reldeg}_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a)$, imagine drawing a tuple $\boldsymbol{u}$ from $\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})$ uniformly at random; then $\mathit{reldeg}_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a)$ is the probability to see $\boldsymbol{u}(A_i) = a$. Given a tuple $\boldsymbol{w}$ over $\mathcal{V}_{i-1}$ and a value $a \in \mathbf{dom}$, define

$$\mathit{reldeg}^*_i(\boldsymbol{w}, a) \quad = \quad \max_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)} \mathit{reldeg}_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a) \tag{8}$$

$$\mathit{constraint}^*_i(\boldsymbol{w}, a) \quad = \quad \arg\max_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)} \mathit{reldeg}_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a). \tag{9}$$

Specifically, $\mathit{constraint}^*_i(\boldsymbol{w}, a)$ returns the constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)$ satisfying the condition $\mathit{reldeg}_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a) = \mathit{reldeg}^*_i(\boldsymbol{w}, a)$. If more than one constraint meets this condition, define $\mathit{constraint}^*_i(\boldsymbol{w}, a)$ to be an arbitrary one among those constraints.

Henceforth, we will fix an arbitrary optimal solution $\{\delta_{\mathcal{Y}|\mathcal{X}} \mid (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}\}$ to the dual modular LP in Section 2. Thus:

$$\begin{aligned}
\prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} N_{\mathcal{Y}|\mathcal{X}}^{\delta_{\mathcal{Y}|\mathcal{X}}} \quad &= \quad \exp_2\Big(\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} \delta_{\mathcal{Y}|\mathcal{X}} \cdot \log N_{\mathcal{Y}|\mathcal{X}}\Big) = \exp_2\Big(\max_{h \in \mathsf{M}_{\mathcal{V}} \cap \mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})\Big) \\
\text{(by (5))} \quad &= \quad \mathit{modular}(\mathsf{DC}) \\
\text{(by Lemma 2)} \quad &= \quad \mathit{polymat}(\mathsf{DC}).
\end{aligned} \tag{10}$$

Finally, for any $i \in [0, k]$ and any tuple $\boldsymbol{w}$ over $\mathcal{V}_i$, define:

$$B_i(\boldsymbol{w}) \quad = \quad \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} \big(\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})\big)^{\delta_{\mathcal{Y}|\mathcal{X}}}. \tag{11}$$

Two observations will be useful later:

- If $i = 0$, then $\boldsymbol{w}$ is a null tuple and $B_0(\text{null}) = \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} (\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})}))^{\delta_{\mathcal{Y}|\mathcal{X}}}$, which is at most $\prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} N_{\mathcal{Y}|\mathcal{X}}^{\delta_{\mathcal{Y}|\mathcal{X}}} = \mathit{polymat}(\mathsf{DC})$.
- If $i = k$ and $\boldsymbol{w} \in \mathit{join}(\mathcal{Q})$, then $R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}$ contains exactly one tuple for any $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$ and thus $B_k(\boldsymbol{w}) = 1$.

Algorithm 1 Our sampling algorithm.

ADC-sample
0. $A_1, A_2, ..., A_k \leftarrow$ a topological order of G_{DC}
1. $\boldsymbol{w}_0 \leftarrow$ a null tuple
2. **for** $i = 1$ to k **do**
3. pick a constraint $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ})$ uniformly at random from $\mathsf{DC}(A_i)$
4. $\boldsymbol{u}^\circ \leftarrow$ a tuple chosen uniformly at random from $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_{i-1})$
/* note: if $i = 1$, then $R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_{i-1} = R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)}$ */
5. $a_i \leftarrow \boldsymbol{u}^\circ(A_i)$
6. **if** $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) \neq \mathit{constraint}^*_{i-1}(\boldsymbol{w}_{i-1}, a_i)$ **then** declare **failure**
7. $\boldsymbol{w}_i \leftarrow$ the tuple over $\mathcal{V}_i$ formed by concatenating $\boldsymbol{w}_{i-1}$ with a_i
8. declare **failure** with probability $1 - p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i)$, where p_{pass} is given in (12)
9. **if** $\boldsymbol{w}_k[F] \in R_F$ for $\forall F \in \mathcal{E}$ **then** /* that is, $\boldsymbol{w}_k \in \mathit{join}(\mathcal{Q})$ */
10. **return** $\boldsymbol{w}_k$

Algorithm. Our sampling algorithm, named ADC-sample, is presented in Algorithm 1. At a high level, it processes one attribute at a time according to the topological order $A_1, A_2, ..., A_k$. The for-loop in Lines 2–9 finds a value a_i for attribute A_i ($i \in [k]$). The algorithm may fail to return anything, but when it *succeeds* (i.e., reaching Line 10), the values $a_1, a_2, ..., a_k$ will make a uniformly random tuple from $\mathit{join}(\mathcal{Q})$.

Next, we explain the details of the for-loop. The loop starts with values $a_1, a_2, ..., a_{i-1}$ already stored in a tuple $\boldsymbol{w}_{i-1}$ (i.e., $\boldsymbol{w}_{i-1}(A_j) = a_j$ for all $j \in [i-1]$). Line 3 randomly chooses a degree constraint $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ})$ from $\mathsf{DC}(A_i)$; see (6). Conceptually, next we identify the main guard $R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)}$ of this constraint, semi-join the relation with $\boldsymbol{w}_{i-1}$, and project the semi-join result on $\mathcal{Y}^\circ$ to obtain $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_{i-1})$. Then, Line 4 randomly chooses a tuple $\boldsymbol{u}^\circ$ from $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_{i-1})$ and Line 5 takes $\boldsymbol{u}^\circ(A_i)$ as the value of a_i (note: $A_i \in \mathcal{Y}^\circ - \mathcal{X}^\circ \subseteq \mathcal{Y}^\circ$). Physically, however, we do not compute $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_{i-1})$ during the sample process. Instead, with proper preprocessing (discussed later), we can acquire the value a_i in $O(1)$ time. Continuing, Line 6 may declare failure and terminate ADC-sample, but if we get past this line, $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ})$ must be exactly $\mathit{constraint}^*_i(\boldsymbol{w}_{i-1}, a_i)$; see (9). As clarified later, the check at Line 6 can be performed in $O(1)$ time. We now form a tuple $\boldsymbol{w}_i$ that takes value a_j on attribute A_j for each $j \in [i]$ (Line 7). Line 8 allows us to pass with probability

$$p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i) = \frac{B_i(\boldsymbol{w}_i)}{B_{i-1}(\boldsymbol{w}_{i-1})} \cdot \frac{1}{\mathit{reldeg}^*_i(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i(A_i))} \tag{12}$$

or otherwise terminate the algorithm by declaring failure. As proved later, $p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i)$ cannot exceed 1 (Lemma 4); moreover, this value can be computed in $O(1)$ time. The overall execution time of ADC-sample is constant.

Analysis. Next we prove that the value in (12) serves as a legal probability value.

▶ **Lemma 4.** *For every $i \in [k]$, we have $p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i) \leq 1$.*

Proof. Consider an arbitrary constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)$. Recall that ADC-sample processes the attributes by the topological order $A_1, ..., A_k$. In the constrained dependency graph G_{DC}, every attribute of $\mathcal{X}$ has an out-going edge to A_i. Hence, all the attributes in $\mathcal{X}$ must be processed prior to A_i. This implies that all the tuples in $R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1}$ must have the same projection on $\mathcal{X}$. Therefore, $\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})$ equals $|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})|$. By the same reasoning, $\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)$ equals $|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)|$. We thus have:

$$\begin{aligned}\frac{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)}{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})} &= \frac{|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)|}{|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})|} \\ &= \frac{\left|\sigma_{A_i=a_i}(\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1}))\right|}{\left|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})\right|} \\ &= reldeg_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}_{i-1}, a_i) \\ &\leq reldeg^*_i(\boldsymbol{w}_{i-1}, a_i). \end{aligned} \tag{13}$$

On the other hand, for any constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \notin \mathsf{DC}(A_i)$, it trivially holds that

$$deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i) \quad \leq \quad deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1}) \tag{14}$$

because $R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i$ is a subset of $R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1}$.

We can now derive

$$\begin{aligned} p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i) &= \frac{1}{reldeg^*_i(\boldsymbol{w}_{i-1}, a_i)} \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}} \Big(\frac{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)}{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})}\Big)^{\delta_{\mathcal{Y}|\mathcal{X}}} \\ \text{(by (14))} \quad &\leq \frac{1}{reldeg^*_i(\boldsymbol{w}_{i-1}, a_i)} \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(A_i)} \Big(\frac{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)}{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})}\Big)^{\delta_{\mathcal{Y}|\mathcal{X}}} \\ \text{(by (13))} \quad &\leq \frac{1}{reldeg^*_i(\boldsymbol{w}_{i-1}, a_i)} \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(A_i)} reldeg^*_i(\boldsymbol{w}_{i-1}, a_i)^{\delta_{\mathcal{Y}|\mathcal{X}}} \\ &= reldeg^*_i(\boldsymbol{w}_{i-1}, a_i)^{\left(\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(A_i)} \delta_{\mathcal{Y}|\mathcal{X}}\right)-1} \leq 1. \end{aligned}$$

The last step used $\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(A_i)} \delta_{\mathcal{Y}|\mathcal{X}} \geq 1$ guaranteed by the dual modular LP. ◀

Next, we argue that every result tuple $\boldsymbol{v} \in join(\mathcal{Q})$ is returned by ADC-sample with the same probability. For this purpose, let us define two random events for each $i \in [k]$:

- event $\mathbf{E1}(i)$: $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) = constraint^*_i(\boldsymbol{w}_{i-1}, \boldsymbol{v}(A_i))$ in the i-th loop of ADC-sample;
- event $\mathbf{E2}(i)$: Line 8 does not declare failure in the i-th loop of ADC-sample.

The probability for ADC-sample to return $\boldsymbol{v}$ can be derived as follows.

$$\begin{aligned} \mathbf{Pr}[\boldsymbol{v} \text{ returned}] &= \prod_{i=1}^{k} \mathbf{Pr}[a_i = \boldsymbol{v}(A_i), \mathbf{E1}(i), \mathbf{E2}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]] \\ &\quad \text{(if } i = 1\text{, then } \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}] \text{ becomes } \boldsymbol{w}_0 = \boldsymbol{v}[\emptyset]\text{, which is vacuously true)} \\ &= \prod_{i=1}^{k} \Big(\mathbf{Pr}[a_i = \boldsymbol{v}(A_i), \mathbf{E1}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]] \cdot \\ &\qquad \mathbf{Pr}[\mathbf{E2}(i) \mid \mathbf{E1}(i), a_i = \boldsymbol{v}(A_i), \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]]\Big). \end{aligned} \tag{15}$$

Observe

$$\begin{aligned} &\mathbf{Pr}[a_i = \boldsymbol{v}(A_i), \mathbf{E1}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]] \\ = \ &\mathbf{Pr}[\mathbf{E1}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]] \cdot \mathbf{Pr}[a_i = \boldsymbol{v}(A_i) \mid \mathbf{E1}(i), \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]] \\ = \ &\frac{1}{|\mathsf{DC}(A_i)|} \cdot \frac{\left|\sigma_{A_i=\boldsymbol{v}(A_i)}(\Pi_{\mathcal{Y}}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{v}[\mathcal{V}_{i-1}]))\right|}{\left|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{v}[\mathcal{V}_{i-1}])\right|} \\ &\text{(note: } (\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) = constraint^*_i(\boldsymbol{v}[\mathcal{V}_{i-1}], \boldsymbol{v}(A_i))\text{, due to } \mathbf{E1}(i) \text{ and } \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]]) \\ = \ &\frac{1}{|\mathsf{DC}(A_i)|} \cdot reldeg_{i,\mathcal{X}^\circ,\mathcal{Y}^\circ}(\boldsymbol{v}[\mathcal{V}_{i-1}], \boldsymbol{v}(A_i)) = \frac{1}{|\mathsf{DC}(A_i)|} \cdot reldeg^*_i(\boldsymbol{v}[\mathcal{V}_{i-1}], \boldsymbol{v}(A_i)). \end{aligned} \tag{16}$$

On the other hand:

$$\begin{aligned} & \mathbf{Pr}[\mathbf{E2}(i) \mid \mathbf{E1}(i), a_i = \boldsymbol{v}(A_i), \boldsymbol{w}_{i-1} = \boldsymbol{v}[\mathcal{V}_{i-1}]] \\ = \; & p_{\text{pass}}(i, \boldsymbol{v}[\mathcal{V}_{i-1}], \boldsymbol{v}[\mathcal{V}_i]) \\ \text{(by (12))} \; = \; & \frac{B_i(\boldsymbol{v}[\mathcal{V}_i])}{B_{i-1}(\boldsymbol{v}[\mathcal{V}_{i-1}])} \cdot \frac{1}{\mathit{reldeg}_i^*(\boldsymbol{v}[\mathcal{V}_{i-1}], \boldsymbol{v}(A_i))}. \end{aligned} \tag{17}$$

Plugging (16) and (17) into (15) yields

$$\begin{aligned} \mathbf{Pr}[\boldsymbol{v} \text{ returned}] \; &= \; \prod_{i=1}^{k} \frac{B_i(\boldsymbol{v}[\mathcal{V}_i])}{B_{i-1}(\boldsymbol{v}[\mathcal{V}_{i-1}])} \cdot \frac{1}{|\mathsf{DC}(A_i)|} = \frac{B_k(\boldsymbol{v}[\mathcal{V}_k])}{B_0(\boldsymbol{v}[\mathcal{V}_0])} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|} \\ &= \; \frac{1}{B_0(\text{null})} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|}. \end{aligned}$$

As the above is identical for every $\boldsymbol{v} \in \mathit{join}(\mathcal{Q})$, we can conclude that each tuple in the join result gets returned by ADC-sample with the same probability. As an immediate corollary, each run of ADC-sample successfully returns a sample from $\mathit{join}(\mathcal{Q})$ with probability

$$\frac{\text{OUT}}{B_0(\text{null})} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|} \geq \frac{\text{OUT}}{\mathit{polymat}(\mathsf{DC})} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|} = \Omega\Big(\frac{\text{OUT}}{\mathit{polymat}(\mathsf{DC})}\Big).$$

In the full version [37], we explain how to preprocess the relations of $\mathcal{Q}$ in $O(\text{IN})$ expected time to ensure that ADC-sample completes in $O(1)$ time.

Performing a Join Sampling Operation. Recall that this operation must either return a uniformly random sample of $\mathit{join}(\mathcal{Q})$ or declare $\mathit{join}(\mathcal{Q}) = \emptyset$. To support this operation, we execute two *threads* concurrently. The first thread repeatedly invokes ADC-sample until it successfully returns a sample. The other thread runs Ngo's algorithm in [29] to compute $\mathit{join}(\mathcal{Q})$ *in full*, after which we can declare $\mathit{join}(\mathcal{Q}) \neq \emptyset$ or sample from $\mathit{join}(\mathcal{Q})$ in constant time. As soon as one thread finishes, we manually terminate the other one.

This strategy guarantees that the join operation completes in $O(\mathit{polymat}(\mathsf{DC})/\max\{1, \text{OUT}\})$ time. To explain why, consider first the scenario where $\text{OUT} \geq 1$. In this case, we expect to find a sample with $O(\mathit{polymat}(\mathsf{DC})/\text{OUT})$ repeats of ADC-sample. Hence, the first thread finishes in $O(\mathit{polymat}(\mathsf{DC})/\text{OUT})$ expected sample time. On the other hand, if $\text{OUT} = 0$, the second thread will finish in $O(\mathit{polymat}(\mathsf{DC}))$ time. This concludes the proof of Theorem 3.

Remarks. When DC has only cardinality constraints (is thus "trivially" acyclic), ADC-sample simplifies into the sampling algorithm of Kim et al. [25]. In retrospect, two main obstacles prevent an obvious extension of their algorithm to an arbitrary acyclic DC. The first is identifying an appropriate way to deal with constraints $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$ where $\mathcal{X} \neq \emptyset$ (such constraints are absent in the degenerated context of [25]). The second obstacle involves determining how to benefit from a topological order (attribute ordering is irrelevant in [25]); replacing the order with a non-topological one may ruin the correctness of ADC-sample.

4 Directed Subgraph Sampling

Given a directed pattern graph $P = (V_P, E_P)$ and a directed data graph $G = (V, E)$, we use $\mathit{occ}(G, P)$ to represent the set of occurrences of P in G. Every vertex in G has an out-degree at most λ. Our goal is to design an algorithm to sample from $\mathit{occ}(G, P)$ efficiently.

Let us formulate the "polymatroid bound" for this problem. Given an integer $m \geq 1$, an integer $\lambda \in [1, m]$, and a pattern $P = (V_P, E_P)$, first build a rule collection $\mathcal{C}$ over V_P as follows: for each edge $(X, Y) \in E_P$, add to $\mathcal{C}$ two rules: $(\emptyset, \{X, Y\}, m)$ and $(\{X\}, \{X, Y\}, \lambda)$. Then, the *directed polymatriod bound* of m, λ, and P can be defined as

$$\mathit{polymat}_{\mathit{dir}}(m, \lambda, P) \quad = \quad \mathit{polymat}(\mathcal{C}) \tag{18}$$

where $\mathit{polymat}(\mathcal{C})$ follows the definition in (4).

This formulation reflects how directed subgraph listing can be processed as a join. Consider a *companion join* $\mathcal{Q}$ constructed from G and P as follows. The schema graph of $\mathcal{Q}$, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is exactly $P = (V_P, E_P)$ (i.e., $\mathcal{V} = V_P$ and $\mathcal{E} = E_P$). For every edge $F = (X, Y) \in E_P$, create a relation $R_F \in \mathcal{Q}$ by inserting, for each edge (x, y) in the data graph G, a tuple $\boldsymbol{u}$ with $\boldsymbol{u}(X) = x$ and $\boldsymbol{u}(Y) = y$ into R_F. The rule collection $\mathcal{C}$ can now be regarded as a set DC of degree constraints with which $\mathcal{Q}$ is consistent, i.e., $\mathcal{Q} \models \mathsf{DC} = \mathcal{C}$. The constraint dependence graph G_{DC} is precisely P. It is immediate that $\mathit{polymat}_{\mathit{dir}}(|E|, \lambda, P) = \mathit{polymat}(\mathsf{DC})$. To find all the occurrences in $\mathit{occ}(G, P)$, it suffices to compute $\mathit{join}(\mathcal{Q})$. Specifically, every tuple $\boldsymbol{u} \in \mathit{join}(\mathcal{Q})$ that uses a distinct value on every attribute in $\mathcal{V}$ $(= V_P)$ matches a unique occurrence in $\mathit{occ}(G, P)$. Conversely, every occurrence in $\mathit{occ}(G, P)$ matches the same number c of tuples in $\mathit{join}(\mathcal{Q})$, where $c \geq 1$ is a constant equal to the number of automorphisms of P. If we denote $\mathrm{OUT} = |\mathit{occ}(G, P)|$ and $\mathrm{OUT}_{\mathcal{Q}} = |\mathit{join}(\mathcal{Q})|$, it follows that $c \cdot \mathrm{OUT} \leq \mathrm{OUT}_{\mathcal{Q}} \leq \mathit{polymat}(\mathsf{DC}) = \mathit{polymat}_{\mathit{dir}}(|E|, \lambda, P)$.

The above observation suggests how directed subgraph sampling can be reduced to join sampling. First, sample a tuple $\boldsymbol{u}$ from $\mathit{join}(\mathcal{Q})$ uniformly at random. Then, check whether $\boldsymbol{u}(A) = \boldsymbol{u}(A')$ for any two distinct attributes $A, A' \in \mathcal{V}$. If so, declare failure; otherwise, declare success and return the unique occurrence matching the tuple $\boldsymbol{u}$. The success probability equals $c \cdot \mathrm{OUT}/\mathrm{OUT}_{\mathcal{Q}}$. In a success event, every occurrence in $\mathit{occ}(G, P)$ has the same probability to be returned.

When P is acyclic, so is G_{DC}, and thus our algorithm in Theorem 3 can be readily applied to handle a subgraph sampling operation. To analyze the performance, consider first $\mathrm{OUT} \geq 1$. We expect to draw $O(\mathrm{OUT}_{\mathcal{Q}}/\mathrm{OUT})$ samples from $\mathit{join}(\mathcal{Q})$ until a success event. As Theorem 3 guarantees retrieving a sample from $\mathit{join}(\mathcal{Q})$ in $O(\mathit{polymat}(\mathsf{DC})/\mathrm{OUT}_{\mathcal{Q}})$ expected time, overall we expect to sample an occurrence from $\mathit{occ}(G, P)$ in

$$O\Big(\frac{\mathit{polymat}(\mathsf{DC})}{\mathrm{OUT}_{\mathcal{Q}}} \cdot \frac{\mathrm{OUT}_{\mathcal{Q}}}{\mathrm{OUT}}\Big) = O\Big(\frac{\mathit{polymat}(\mathsf{DC})}{\mathrm{OUT}}\Big)$$

time. To prepare for the possibility of $\mathrm{OUT} = 0$, we apply the "two-thread approach" in Section 3. We run a concurrent thread that executes Ngo's algorithm in [29], which finds the whole $\mathit{join}(\mathcal{Q})$, and hence $\mathit{occ}(G, P)$, in $O(\mathit{polymat}(\mathsf{DC}))$ time, after which we can declare $\mathit{occ}(G, P) = \emptyset$ or sample from $\mathit{occ}(G, P)$ in $O(1)$ time. By accepting whichever thread finishes earlier, we ensure that the operation completes in $O(\mathit{polymat}(\mathsf{DC})/\max\{1, \mathrm{OUT}\})$ time.

The main challenge arises when P is cyclic. In this case, G_{DC} (which equals P) is cyclic. Thus, DC becomes a cyclic set of degree constraints, rendering neither Theorem 3 nor Ngo's algorithm in [29] applicable. We overcome this challenge with the lemma below.

▶ **Lemma 5.** *If DC is cyclic, we can always find an acyclic subset $\mathsf{DC}' \subset \mathsf{DC}$ satisfying $\mathit{polymat}(\mathsf{DC}') = \Theta(\mathit{polymat}(\mathsf{DC}))$.*

The proof is presented in Appendix A. Because $\mathcal{Q} \models \mathsf{DC}$ and DC' is a subset of DC, we know that $\mathcal{Q}$ must be consistent with DC' as well, i.e., $\mathcal{Q} \models \mathsf{DC}'$. Therefore, our Theorem 3 can now be used to extract a sample from $\mathit{join}(\mathcal{Q})$ in $O(\mathit{polymat}_{\mathit{dir}}(\mathsf{DC}')/\max\{1, \mathrm{OUT}_{\mathcal{Q}}\})$

time. Importantly, Lemma 5 also permits us to directly apply Ngo's algorithm in [29] to compute $join(\mathcal{Q})$ in $O(polymat(\mathsf{DC}'))$ time. Therefore, we can now apply the two-thread technique to sample from $occ(G, P)$ in

$$O\Big(\frac{polymat(\mathsf{DC}')}{\max\{1, \mathrm{OUT}\}}\Big) = O\Big(\frac{polymat(\mathsf{DC})}{\max\{1, \mathrm{OUT}\}}\Big) = O\Big(\frac{polymat_{dir}(|E|, \lambda, P)}{\max\{1, \mathrm{OUT}\}}\Big)$$

time. We thus have arrived yet:

▶ **Theorem 6.** *Let $G = (V, E)$ be a simple directed data graph, where each vertex has an out-degree at most λ. Let $P = (V_P, E_P)$ be a simple weakly-connected directed pattern graph with a constant number of vertices. We can build in $O(|E|)$ expected time a data structure that supports each subgraph sampling operation in $O(polymat_{dir}(|E|, \lambda, P)/\max\{1, \mathrm{OUT}\})$ expected time, where* OUT *is the number of occurrences of P in G, and $polymat_{dir}(|E|, \lambda, P)$ is the directed polymatrioid bound in* (18).

Remarks. For subgraph *listing*, Jayaraman et al. [18] presented a sophisticated method that also enables the application of Ngo's algorithm in [29] to a cyclic P. Given the companion join $\mathcal{Q}$, they employ the *degree uniformization technique* [20] to generate $t = O(\mathrm{polylog}\,|E|)$ new joins $\mathcal{Q}_1, \mathcal{Q}_2, ..., \mathcal{Q}_t$ such that $join(\mathcal{Q}) = \bigcup_{i=1}^{t} join(\mathcal{Q}_i)$. For each $i \in [t]$, they construct an acyclic set DC_i of degree constraints (which is not always a subset of DC) with the property $\sum_{i=1}^{t} polymat(\mathsf{DC}_i) \leq polymat(\mathsf{DC})$. Each join $\mathcal{Q}_i$ ($i \in [t]$) can then be processed by Ngo's algorithm in $O(polymat(\mathsf{DC}_i))$ time, thus giving an algorithm for computing $join(\mathcal{Q})$ (and hence $occ(G, P)$) in $O(polymat(\mathsf{DC}))$ time. On the other hand, Lemma 5 facilitates a *direct* application of Ngo's algorithm to $\mathcal{Q}$, implying the non-necessity of degree uniformization in subgraph listing. We believe that this simplification is noteworthy and merits its own dedicated exposition, considering the critical nature of the subgraph listing problem. In the absence of Lemma 5, integrating our join-sampling algorithm with the methodology of [18] for the purpose of subgraph sampling would require substantially more effort. Our proof of Lemma 5 *does* draw upon the analysis of [18], as discussed in depth in Appendix A.

5 Concluding Remarks

Our new sampling algorithms imply new results on several other fundamental problems. We will illustrate this with respect to evaluating a join $\mathcal{Q}$ consistent with an acyclic set DC of degree constraints. Similar implications also apply to subgraph sampling.

- By standard techniques [10, 13], we can estimate the output size OUT up to a relative error ϵ with high probability (i.e., at least $1 - 1/\mathrm{IN}^c$ for an arbitrarily large constant c) in time $\tilde{O}(\frac{1}{\epsilon^2}\frac{polymat(\mathsf{DC})}{\max\{1,\mathrm{OUT}\}})$ after a preprocessing of $O(\mathrm{IN})$ expected time.
- Employing a technique in [13], we can, with high probability, report all the tuples in $join(\mathcal{Q})$ with a delay of $\tilde{O}(\frac{polymat(\mathsf{DC})}{\max\{1,\mathrm{OUT}\}})$. In this context, "delay" refers to the maximum interval between the reporting of two successive result tuples, assuming the presence of a placeholder tuple at the beginning and another at the end.
- In addition to the delay guarantee, our algorithm in the second bullet can, with high probability, report the tuples of $join(\mathcal{Q})$ in a random permutation. This means that each of the OUT! possible permutations has an equal probability of being the output.

All of the results presented above compare favorably with the current state of the art as presented in [13]. This is primarily due to the superiority of $polymat(\mathsf{DC})$ over $AGM(\mathcal{Q})$. In addition, our findings in the last two bullet points also complement Ngo's algorithm as described in [29] in a satisfying manner.

References

1 Amir Abboud, Seri Khoury, Oree Leibowitz, and Ron Safier. Listing 4-cycles. *CoRR*, abs/2211.10022, 2022. doi:10.48550/arXiv.2211.10022.

2 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

3 Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 275–286, 1999. doi:10.1145/304182.304207.

4 Noga Alon. On the number of subgraphs of prescribed type of graphs with a given number of edges. *Israel Journal of Mathematics*, 38:116–130, 1981.

5 Kaleb Alway, Eric Blais, and Semih Salihoglu. Box covers and domain orderings for beyond worst-case join processing. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 3:1–3:23, 2021. doi:10.4230/LIPIcs.ICDT.2021.3.

6 Albert Atserias, Martin Grohe, and Daniel Marx. Size bounds and query plans for relational joins. *SIAM Journal on Computing*, 42(4):1737–1767, 2013. doi:10.1137/110859440.

7 Matthias Bentert, Till Fluschnik, Andre Nichterlein, and Rolf Niedermeier. Parameterized aspects of triangle enumeration. *Journal of Computer and System Sciences (JCSS)*, 103:61–77, 2019. doi:10.1016/j.jcss.2019.02.004.

8 Andreas Bjorklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 223–234, 2014. doi:10.1007/978-3-662-43948-7_19.

9 Surajit Chaudhuri, Rajeev Motwani, and Vivek R. Narasayya. On random sampling over joins. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 263–274, 1999. doi:10.1145/304182.304206.

10 Yu Chen and Ke Yi. Random sampling and size estimation over cyclic joins. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 7:1–7:18, 2020. doi:10.4230/LIPIcs.ICDT.2020.7.

11 N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal of Computing*, 14(1):210–223, 1985. doi:10.1137/0214017.

12 Kyle Deeds, Dan Suciu, Magda Balazinska, and Walter Cai. Degree sequence bound for join cardinality estimation. In *Proceedings of International Conference on Database Theory (ICDT)*, volume 255, pages 8:1–8:18, 2023. doi:10.4230/LIPIcs.ICDT.2023.8.

13 Shiyuan Deng, Shangqi Lu, and Yufei Tao. On join sampling and the hardness of combinatorial output-sensitive join algorithms. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 99–111, 2023. doi:10.1145/3584372.3588666.

14 David Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.*, 3(3):1–27, 1999. doi:10.7155/jgaa.00014.

15 Hendrik Fichtenberger, Mingze Gao, and Pan Peng. Sampling arbitrary subgraphs exactly uniformly in sublinear time. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 45:1–45:13, 2020. doi:10.4230/LIPIcs.ICALP.2020.45.

16 Tomasz Gogacz and Szymon Torunczyk. Entropy bounds for conjunctive queries with functional dependencies. In *Proceedings of International Conference on Database Theory (ICDT)*, volume 68, pages 15:1–15:17, 2017. doi:10.4230/LIPIcs.ICDT.2017.15.

17 Chinh T. Hoang, Marcin Kaminski, Joe Sawada, and R. Sritharan. Finding and listing induced paths and cycles. *Discrete Applied Mathematics*, 161(4-5):633–641, 2013. doi:10.1016/j.dam.2012.01.024.

18 Sai Vikneshwar Mani Jayaraman, Corey Ropell, and Atri Rudra. Worst-case optimal binary join algorithms under general ℓ_p constraints. *CoRR*, abs/2112.01003, 2021. doi:10.48550/arXiv.2112.01003.

19 Ce Jin and Yinzhan Xu. Removing additive structure in 3sum-based reductions. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 405–418, 2023. doi:10.1145/3564246.3585157.

20 Manas Joglekar and Christopher Re. It's all a matter of degree - using degree information to optimize multiway joins. *Theory Comput. Syst.*, 62(4):810–853, 2018. doi:10.1007/s00224-017-9811-8.

21 Mahmoud Abo Khamis, Vasileios Nakos, Dan Olteanu, and Dan Suciu. Join size bounds using lp-norms on degree sequences. *CoRR*, abs/2306.14075, 2023. doi:10.48550/arXiv.2306.14075.

22 Mahmoud Abo Khamis, Hung Q. Ngo, Christopher Re, and Atri Rudra. Joins via geometric resolutions: Worst case and beyond. *ACM Transactions on Database Systems (TODS)*, 41(4):22:1–22:45, 2016. doi:10.1145/2967101.

23 Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. Computing join queries with functional dependencies. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 327–342, 2016. doi:10.1145/2902251.2902289.

24 Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 429–444, 2017. doi:10.1145/3034786.3056105.

25 Kyoungmin Kim, Jaehyun Ha, George Fletcher, and Wook-Shin Han. Guaranteeing the $\tilde{O}$(AGM/OUT) runtime for uniform sampling and size estimation over joins. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 113–125, 2023. doi:10.1145/3584372.3588676.

26 George Manoussakis. Listing all fixed-length simple cycles in sparse graphs in optimal time. In *Fundamentals of Computation Theory*, pages 355–366, 2017. doi:10.1007/978-3-662-55751-8_28.

27 Gonzalo Navarro, Juan L. Reutter, and Javiel Rojas-Ledesma. Optimal joins using compact data structures. In *Proceedings of International Conference on Database Theory (ICDT)*, volume 155, pages 21:1–21:21, 2020. doi:10.4230/LIPIcs.ICDT.2020.21.

28 Jaroslav Nesetril and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

29 Hung Q. Ngo. Worst-case optimal join algorithms: Techniques, results, and open problems. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 111–124, 2018. doi:10.1145/3196959.3196990.

30 Hung Q. Ngo, Dung T. Nguyen, Christopher Re, and Atri Rudra. Beyond worst-case analysis for joins with minesweeper. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 234–245, 2014. doi:10.1145/2594538.2594547.

31 Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-Case Optimal Join Algorithms: [Extended Abstract]. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 37–48, 2012. doi:10.1145/2213556.2213565.

32 Hung Q. Ngo, Ely Porat, Christopher Re, and Atri Rudra. Worst-case optimal join algorithms. *Journal of the ACM (JACM)*, 65(3):16:1–16:40, 2018. doi:10.1145/3180143.

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

34 Dan Suciu. Applications of information inequalities to database theory problems. *CoRR*, abs/2304.11996, 2023. doi:10.48550/arXiv.2304.11996.

35 Maciej M. Syslo. An efficient cycle vector space algorithm for listing all cycles of a planar graph. *SIAM Journal of Computing*, 10(4):797–808, 1981. doi:10.1137/0210062.

36 Todd L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 96–106, 2014. doi:10.5441/002/icdt.2014.13.

37 Ru Wang and Yufei Tao. Join sampling under acyclic degree constraints and (cyclic) subgraph sampling, 2023. doi:10.48550/arXiv.2312.12797.

38 Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. Random sampling over joins revisited. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 1525–1539, 2018. doi:10.1145/3183713.3183739.

A Proof of Lemma 5

Let us rephrase the problem as follows. Let $P = (V_P, E_P)$ be a *cyclic* pattern graph. Given an integer $m \geq 1$ and an integer $\lambda \in [1, m]$, define DC to be a set of degree constraints over V_P that contains two constraints for each edge $(X, Y) \in E_P$: $(\emptyset, \{X, Y\}, m)$ and $(\{X\}, \{X, Y\}, \lambda)$. The constraint dependence graph G_{DC} is exactly P and, hence, is cyclic. We want to prove the existence of an acyclic $\mathsf{DC}' \subset \mathsf{DC}$ such that $\mathit{polymat}(\mathsf{DC}') = \mathit{polymat}(\mathsf{DC})$. We will first tackle the situation where $\lambda > \sqrt{m}$ before proceeding to the opposite scenario. The former case presents a more intriguing line of argumentation than the latter.

A.1 Case $\lambda > \sqrt{m}$

For every edge $(X, Y) \in G_{\mathsf{DC}} = (V_P, E_P)$, define two variables: $x_{X,Y}$ and $z_{X,Y}$. Jayaraman et al. [18] showed that, for $\lambda > \sqrt{m}$, $\mathit{polymat}(\mathsf{DC})$ is, up to a constant factor, the optimal value of the following LP (named $\mathrm{LP}^{(+)}$ following [18]):

$$\mathbf{LP}^{(+)}\ [18] \quad \min \sum_{(X,Y)\in E_P} x_{X,Y} \log m + z_{X,Y} \log \lambda \text{ subject to}$$

$$\sum_{(X,A)\in E_P} (x_{X,A} + z_{X,A}) + \sum_{(A,Y)\in E_P} x_{A,Y} \geq 1 \qquad \forall A \in V_P$$

$$x_{X,Y} \geq 0,\ z_{X,Y} \geq 0 \qquad \forall (X, Y) \in E_P$$

▶ **Lemma 7.** *There exists an optimal solution to $LP^{(+)}$ satisfying the condition that the edges in $\{(X, Y) \in E_P \mid z_{X,Y} > 0\}$ induce an acyclic subgraph of G_{DC}.*

We note that while the above lemma is not expressly stated in [18], it can be extrapolated from the analysis presented in Section H.2 of [18]. Nevertheless, the argument laid out in [18] is quite intricate. Our proof, which will be presented below, incorporates news ideas beyond their argument and is considerably shorter. Specifically, these new ideas are evidenced in the way we formulate a novel LP optimal solution in (19)-(22).

Proof of Lemma 7. Consider an arbitrary optimal solution to $\mathrm{LP}^{(+)}$ that sets $x_{X,Y} = x^*_{X,Y}$ and $z_{X,Y} = z^*_{X,Y}$ for each $(X, Y) \in E_P$. If the edge set $\{(X, Y) \in E_P \mid z^*_{X,Y} > 0\}$ induces an acyclic graph, we are done. Next, we consider that G_{DC} contains a cycle.

Suppose that (A_1, A_2) is the edge in the cycle with the smallest $z^*_{A_1,A_2}$ (breaking ties arbitrarily). Let (A_2, A_3) be the edge succeeding (A_1, A_2) in the cycle. It thus follows that $z^*_{A_2,A_3} \geq z^*_{A_1,A_2}$. Define

$$x'_{A_2,A_3} = x^*_{A_2,A_3} + z^*_{A_1,A_2} \tag{19}$$

$$x'_{A_1,A_2} = x^*_{A_1,A_2} \tag{20}$$

$$z'_{A_2,A_3} = 0 \tag{21}$$

$$z'_{A_1,A_2} = 0 \tag{22}$$

For every edge $(X, Y) \in E_P \setminus \{(A_1, A_2), (A_2, A_3)\}$, set $x'_{X,Y} = x^*_{X,Y}$ and $z'_{X,Y} = z^*_{X,Y}$. It is easy to verify that, for every vertex $A \in V_P$, we have

$$\sum_{(X,A)\in E_P} (x'_{X,A} + z'_{X,A}) + \sum_{(A,Y)\in E_P} x'_{A,Y} \geq \sum_{(X,A)\in E_P} (x^*_{X,A} + z^*_{X,A}) + \sum_{(A,Y)\in E_P} x^*_{A,Y}.$$

Therefore, $\{x'_{X,Y}, z'_{X,Y} \mid (X, Y) \in E_P\}$ serves as a feasible solution to $\mathrm{LP}^{(+)}$. However:

$$\begin{aligned}
&\Big(\sum_{(X,Y)\in E_P} x'_{X,Y}\log m + z'_{X,Y}\log\lambda\Big) - \Big(\sum_{(X,Y)\in E_P} x^*_{X,Y}\log m + z^*_{X,Y}\log\lambda\Big)\\
=\ & z^*_{A_1,A_2}\log m - (z^*_{A_1,A_2} + z^*_{A_2,A_3})\log\lambda\\
\leq\ & z^*_{A_1,A_2}\log m - 2\cdot z^*_{A_1,A_2}\log\lambda\\
<\ & 0 \qquad (23)
\end{aligned}$$

where the last step used the fact $\lambda^2 > m$. This contradicts the optimality of $\{x^*_{X,Y}, z^*_{X,Y} \mid (X,Y)\in E_P\}$. ◀

We now build a set DC' of degree constraints as follows. First, take an optimal solution $\{x^*_{X,Y}, z^*_{X,Y} \mid (X,Y)\in E_P\}$ to $\mathrm{LP}^{(+)}$ promised by Lemma 7. Add to DC' a constraint $(X,\{X,Y\},\lambda)$ for every $(X,Y)\in E_P$ satisfying $z^*_{X,Y} > 0$. Then, for every edge $(X,Y)\in E_P$, add to DC' a constraint $(\emptyset,\{X,Y\},m)$. The DC' thus constructed must be acyclic. Denote by $G_{\mathsf{DC}'} = (V'_P, E'_P)$ the degree constraint graph of DC'. Note that $V_P = V'_P$ and $E'_P \subset E_P$.

▶ **Lemma 8.** *The* DC' *constructed in the above manner satisfies* $\mathit{polymat}(\mathsf{DC}') = \Theta(\mathit{polymat}(\mathsf{DC}))$.

Proof. We will first establish $\mathit{polymat}(\mathsf{DC}') \geq \mathit{polymat}(\mathsf{DC})$. Remember that $\mathit{polymat}(\mathsf{DC}')$ is the optimal value of the modular LP (in its primal form) defined by DC', as described in Section 2. Similarly, $\mathit{polymat}(\mathsf{DC})$ is the optimal value of the modular LP defined by DC. Given that $\mathsf{DC}' \subset \mathsf{DC}$, the LP defined by DC' incorporates only a subset of the constraints found in the LP defined by DC. Therefore, it must be the case that $\mathit{polymat}(\mathsf{DC}') \geq \mathit{polymat}(\mathsf{DC})$.

The rest of the proof will show $\mathit{polymat}(\mathsf{DC}') = O(\mathit{polymat}(\mathsf{DC}))$, which will establish the lemma. Consider the following LP:

$$\begin{aligned}
\mathbf{LP}_1^{(+)} \qquad & \min \sum_{(X,Y)\in E_P} x_{X,Y}\log m + z_{X,Y}\log\lambda \text{ subject to}\\
& \sum_{(X,A)\in E_P} x_{X,A} + \sum_{(A,Y)\in E_P} x_{A,Y} + \sum_{(X,A)\in E'_P} z_{X,A} \geq 1 \qquad \forall A\in\mathcal{V}_P\\
& x_{X,Y}\geq 0,\ z_{X,Y}\geq 0 \qquad \forall (X,Y)\in E_P
\end{aligned}$$

The condition $(X,A)\in E'_P$ in the first inequality marks the difference between $\mathrm{LP}_1^{(+)}$ and $\mathrm{LP}^{(+)}$. Note that the two LPs have the same objective function.

▷ Claim 1. $\mathrm{LP}_1^{(+)}$ and $\mathrm{LP}^{(+)}$ have the same optimal value.

To prove the claim, first observe that any feasible solution $\{x_{X,Y}, z_{X,Y} \mid (X,Y)\in E_P\}$ to $\mathrm{LP}_1^{(+)}$ is also a feasible solution to $\mathrm{LP}^{(+)}$. Hence, the optimal value of $\mathrm{LP}^{(+)}$ cannot exceed that of $\mathrm{LP}_1^{(+)}$. On the other hand, recall that earlier we have identified an optimal solution $\{x^*_{X,Y}, z^*_{X,Y} \mid (X,Y)\in E_P\}$ to $\mathrm{LP}^{(+)}$. By how DC' is built from that solution and how $G_{\mathsf{DC}'} = (V'_P, E'_P)$ is built from DC', it must hold that $z^*_{X,Y} = 0$ for every $(X,Y)\in E_P \setminus E'_P$. Hence, $\{x^*_{X,Y}, z^*_{X,Y} \mid (X,Y)\in E_P\}$ makes a feasible solution to $\mathrm{LP}_1^{(+)}$. This implies that $\{x^*_{X,Y}, z^*_{X,Y} \mid (X,Y)\in E_P\}$ must be an optimal solution to $\mathrm{LP}_1^{(+)}$. Claim 1 now follows.

Consider another LP:

$$\mathbf{LP_2^{(+)}} \qquad \min \sum_{(X,Y)\in E_P} x_{X,Y}\log m + \sum_{(X,Y)\in E'_P} z_{X,Y}\log\lambda \text{ subject to}$$

$$\sum_{(X,A)\in E_P} x_{X,A} + \sum_{(A,Y)\in E_P} x_{A,Y} + \sum_{(X,A)\in E'_P} z_{X,A} \geq 1 \qquad \forall A \in \mathcal{V}_P$$

$$x_{X,Y} \geq 0 \qquad \forall (X,Y) \in E_P$$

$$z_{X,Y} \geq 0 \qquad \forall (X,Y) \in E'_P$$

$\mathrm{LP}_2^{(+)}$ differs from $\mathrm{LP}_1^{(+)}$ in that the former drops the variables $z_{X,Y}$ of those edges $(X,Y) \in E_P \setminus E'_P$. This happens both in the constraints and the objective function.

▷ Claim 2. $\mathrm{LP}_1^{(+)}$ and $\mathrm{LP}_2^{(+)}$ have the same optimal value.

To prove the claim, first observe that, given a feasible solution $\{x_{X,Y} \mid (X,Y) \in E_P\} \cup \{z_{X,Y} \mid (X,Y) \in E'_P\}$ to $\mathrm{LP}_2^{(+)}$, we can extend it into a feasible solution to $\mathrm{LP}_1^{(+)}$ by padding $Z_{X,Y} = 0$ for each $(X,Y) \in E_P \setminus E'_P$. Hence, the optimal value of $\mathrm{LP}_1^{(+)}$ cannot exceed that of $\mathrm{LP}_2^{(+)}$. On the other hand, as mentioned before, $\{x^*_{X,Y}, z^*_{X,Y} \mid (X,Y) \in E_P\}$ is an optimal solution to $\mathrm{LP}_1^{(+)}$. In this solution, $z^*_{X,Y} = 0$ for every $(X,Y) \in E_P \setminus E'_P$. Thus, $\{x^*_{X,Y} \mid (X,Y) \in E_P\} \cup \{z^*_{X,Y} \mid (X,Y) \in E'_P\}$ makes a feasible solution to $\mathrm{LP}_2^{(+)}$, achieving the same objective function value as the optimal value of $\mathrm{LP}_1^{(+)}$. Claim 2 now follows.

Finally, notice that $\mathrm{LP}_2^{(+)}$ is exactly the dual modular LP defined by DC'. Hence, $\log(\mathit{polymat}(\mathsf{DC}'))$ is exactly the optimal value of $\mathrm{LP}_2^{(+)}$. Thus, $\mathit{polymat}(\mathsf{DC}') = O(\mathit{polymat}(\mathsf{DC}))$ can now be derived from the above discussion and the fact that $\log(\mathit{polymat}(\mathsf{DC}))$ is asymptotically the optimal value of $\mathrm{LP}^{(+)}$. ◀

A.2 Case $\lambda \leq \sqrt{m}$

Let us first define several concepts. A *directed star* refers to a directed graph where there are $t \geq 2$ vertices, among which one vertex, designated the *center*, has $t-1$ edges (in-coming and out-going edges combined), and every other vertex, called a *petal*, has only one edge (which can be an in-coming or out-going edge). Now, consider a directed bipartite graph between U_1 and U_2, each being an independent sets of vertices (an edge may point from one vertex in U_1 to a vertex in U_2, or vice versa). A *directed star cover* of the bipartite graph is a set of directed stars such that

- each directed star is a subgraph of the bipartite graph,
- no two directed stars share a common edge, and
- every vertex in $U_1 \cup U_2$ appears in exactly one directed star.

A directed star cover is *minimum* if it has the least number of edges, counting all directed stars in the cover.

Next, we review an expression about $\mathit{polymat}(\mathsf{DC})$ derived in [18]. Find all the strongly connected components (SCCs) of $G_{\mathsf{DC}} = (V_P, E_P)$. Adopting terms from [18], an SCC is classified as (i) a *source* if it has no in-coming edge from another SCC, or a *non-source* otherwise; (ii) *trivial* if it consists of a single vertex, or *non-trivial* otherwise. Define:

- S = the set of vertices in G_{DC} each forming a trivial source SCC by itself.
- T = the set of vertices in G_{DC} receiving an in-coming edge from at least one vertex in S.

Take a minimum directed star cover of the directed bipartite graph induced by S and T. Define

- S_1 = the set of vertices in S each serving as the center of some directed star in the cover.
- $S_2 = S \setminus S_1$.

- T_2 = the set of vertices in T each serving as the center of some directed star in the cover.
- $T_1 = T \setminus T_2$.

Note that the meanings of the symbols S_1, S_2, T_1, and T_2 follow exactly those in [18] for the reader's convenience (in particular, note the semantics of T_1 and T_2).

We now introduce three quantities:

- c_1: the number of non-trivial source SCCs;
- n_1: the total number of vertices in non-trivial source SCCs;
- $n_2 = |V_P| - n_1 - |S| - |T|$.

Jayaraman et al. [18] showed:

$$polymat_{dir}(m, \lambda, P) \quad = \quad \Theta\Big(m^{c_1+|S|} \cdot \lambda^{n_1+n_2+|T_1|-2c_1-|S_1|}\Big). \tag{24}$$

Let $G'_{\mathsf{DC}} = (V'_P, E'_P)$ be an arbitrary weakly-connected acyclic subgraph of G_{DC} satisfying all the conditions below.

- $V_P = V'_P$.
- E'_P contains all the edges in the minimum directed star cover identified earlier.
- In each non-trivial source SCC, every vertex, except for one, has one in-coming edge included in E'_P. We will refer to the vertex X with no in-coming edges in E'_P as the SCC's *root*. The fact that every other vertex Y in the SCC has an in-coming edge in E'_P implies $(X, Y) \in E'_P$ for at least one Y. We designate one such (X, Y) as the SCC's *main edge*.
- In each non-trivial non-source SCC, every vertex has an in-coming edge included in E'_P.

It is rudimentary to verify that such a subgraph G'_{DC} must exist.

From $G_{\mathsf{DC}} = (V_P, E_P)$ and $G'_{\mathsf{DC}} = (V'_P, E'_P)$, we create a set DC' of degree constraints as follows.

- For each edge $(X, Y) \in E_P$ (note: not E'_P), add a constraint $(\emptyset, \{X, Y\}, m)$ to DC'.
- We inspect each directed star in the minimum directed star cover and distinguish two possibilities.
  - Scenario 1: The star's center X comes from S_1. Let the star's petals be $Y_1, Y_2, ..., Y_t$ for some $t \geq 1$; the ordering of the petals does not matter. For each $i \in [t-1]$, we add a constraint $(\{X\}, \{X, Y_i\}, \lambda)$ to DC'. We will refer to (X, Y_t) as the star's *main edge*.
  - Scenario 2: The star's center X comes from T_2. Nothing needs to be done.
- Consider now each non-trivial source SCC. Remember that every vertex Y, other than the SCC's root, has an in-coming edge $(X, Y) \in E'_P$. For every such Y, if (X, Y) is not the SCC's main edge, add a constraint $(\{X\}, \{X, Y\}, \lambda)$ to DC'.
- Finally, we examine each non-source SCC. As mentioned, every vertex Y in such an SCC has an in-coming edge $(X, Y) \in E'_P$. For every Y, add a constraint $(\{X\}, \{X, Y\}, \lambda)$ to DC'.

The rest of the proof will show $polymat(\mathsf{DC}') = \Theta(polymat(\mathsf{DC}))$. As $\mathsf{DC}' \subset \mathsf{DC}$, we must have $polymat(\mathsf{DC}') \geq polymat(\mathsf{DC})$ following the same reasoning used in the $\lambda > \sqrt{m}$ case.

We will now proceed to argue that $polymat(\mathsf{DC}') = O(polymat(\mathsf{DC}))$. Recall that $\log(polymat(\mathsf{DC}'))$ is the optimal value of the dual modular LP of DC' (see Section 2). On the other hand, the value of $polymat(\mathsf{DC})$ satisfies (24). In the following, we will construct a feasible solution to the dual modular LP of DC' under which the LP's objective function achieves the value of

$$\Big((c_1 + |S|) \cdot \log m\Big) + (n_1 + n_2 + |T_1| - 2c_1 - |S_1|) \cdot \log \lambda \tag{25}$$

which will be sufficient for proving Lemma 7.

The dual modular LP associates every constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}'$ with a variable $\delta_{\mathcal{Y}|\mathcal{X}}$. We determine these variables' values as follows.

- For every constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}'$ where $N_{\mathcal{Y}|\mathcal{X}} = \lambda$, set $\delta_{\mathcal{Y}|\mathcal{X}} = 1$.
- Consider each directed star in the minimum directed star.
  - Scenario 1: The star's center X comes from S_1. For the star's main edge (X, Y), the constraint $(\emptyset, \{X, Y\}, m)$ exists in DC'. Set $\delta_{\{X,Y\}|\emptyset} = 1$.
  - Scenario 2: The star's center X comes from T_2. For every petal Y of the star, the constraint $(\emptyset, \{X, Y\}, m)$ exists in DC'. Set $\delta_{\{X,Y\}|\emptyset} = 1$.
- Consider each non-trivial source SCC. Let (X, Y) be the main edge of the SCC. The constraint $(\emptyset, \{X, Y\}, m)$ exists in DC'. Set $\delta_{\{X,Y\}|\emptyset} = 1$.

The other variables that have not yet been mentioned are all set to 0.

It is tedious but straightforward to verify that all the constraints of the dual modular LP are fulfilled. To confirm that the objective function indeed evaluates to (25), observe:

- There are $c_1 + |S|$ constraints of the form $(\emptyset, \{X, Y\}, m)$ with $\delta_{\{X,Y\}|\emptyset} = 1$. Specifically, c_1 of them come from the roots of the non-trivial source SCCs, $|S_1|$ of them come from the star center vertices in S_1, and $|S_2|$ of them come from the petal vertices in S_2.
- There are $n_1 + n_2 + |T_1| - 2c_1 - |S_1|$ of the form $(\{X\}, \{X, Y\}, \lambda)$ with $\delta_{\{X,Y\}|\{X\}} = 1$. Specifically, $n_1 - 2c_1$ of them come from the non-main edges of the non-trivial source SCCs, n_2 of them come from the vertices that are not in any non-trivial source SCC and are not in $S \cup T$, and $|T_1| - |S_1|$ of them come from the petal vertices that are (i) in T_1 but (ii) not in the main edges of their respective stars.

We now conclude the whole proof of Lemma 5.