# Join and Subgraph Sampling under Degree Constraints

Ru Wang Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
{*rwang21*, *taoyf*}*@cse.cuhk.edu.hk*

April 28, 2025

**Abstract**

This article studies two problems related to sampling from the results of database queries. The first one is to uniformly sample a tuple from the result of a join obeying an acyclic set of degree constraints (the join itself need not be acyclic). The second is to uniformly sample a given subgraph pattern's occurrence (the pattern may contain cycles) in a directed data graph. It is shown that, after a linear expected-time preprocessing, both problems admit an algorithm drawing a sample in $O(\mathit{polymat}/\max\{1, \text{OUT}\})$ expected time, where OUT and *polymat* are the "full result size" and "polymatroid bound" of the underlying problem, respectively (assuming data complexity). These results are derived with a new sampling algorithm for the former problem and a new graph-theoretic theorem for the latter.

A preliminary version of this article appeared in ICDT'24
This work was supported in part by GRF projects 14207820, 14203421, and 14222822 from HKRGC.

# 1 Introduction

(Natural) joins in relational database systems are known to be computation-intensive, with their costs surging drastically in response to growing 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 query optimization can operate effectively with random samples (see, e.g., [25, 34, 36]). 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 identify 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 article will study *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 $\mathcal{X}$ 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 $schema(R)$. We say that $R$ is a *binary* relation if $|schema(R)| = 2$.

For 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 $\Pi$ is the standard "projection" operator in relational algebra. If in addition $\mathcal{Y} = schema(R)$, then $deg_{\mathcal{Y}|\mathcal{X}}(R) = |R|$.

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 $\mathcal{Q}$ has a constant number of relations and $schema(\mathcal{Q})$ has a constant size. 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 $\mathsf{DC}$ 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 $\mathsf{DC}$ — written as $\mathcal{Q} \models \mathsf{DC}$ — if every degree constraint in $\mathsf{DC}$ is guarded by at least one relation in $\mathcal{Q}$. It is safe to assume that $\mathsf{DC}$ 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 $\mathsf{DC}$.

In this work, we concentrate on "acyclic" degree constraints. To formalize this notion, let us define a *constraint dependency graph* $G_{\mathsf{DC}}$ as follows. This is a directed graph whose vertex set is $schema(\mathcal{Q})$ (i.e., each vertex of $G_{\mathsf{DC}}$ is an attribute in $schema(\mathcal{Q})$). 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 $G_{\mathsf{DC}}$ for every pair $(X, Y) \in \mathcal{X} \times (\mathcal{Y} \setminus \mathcal{X})$. We say that the set $\mathsf{DC}$ is *acyclic* if $G_{\mathsf{DC}}$ is an acyclic graph; otherwise, $\mathsf{DC}$ 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 $\mathcal{X} = \emptyset$, $\mathcal{Y} = schema(R)$, and $N_{\mathcal{Y}|\mathcal{X}} = |R|$. Such a constraint — known as a *cardinality constraint* — is assumed to be always present in $\mathsf{DC}$. As all cardinality constraints have $\mathcal{X} = \emptyset$, they do not affect the construction of $G_{\mathsf{DC}}$. Consequently, if $\mathsf{DC}$ only contains cardinality constraints, then $G_{\mathsf{DC}}$ has no edges and hence trivially acyclic. Moreover, readers should avoid the misconception that "an acyclic $G_{\mathsf{DC}}$ 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) Subgraph 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 $u_1, u_2 \in V_{sub}$, an edge $(u_1, u_2)$ exists 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 $\lambda$.

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

### 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 $join(\mathcal{Q})$. 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

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

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 $X \in \mathcal{V}$, we have $\sum_{F \in \mathcal{E}: X \in F} w(F) \geq 1$ (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 $AGM(\mathcal{Q}) = \min_w \prod_{F \in \mathcal{E}} |R_F|^{w(F)}$, where the minimization is over all the functional edge covers $w$ of $\mathcal{G}$.

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 [13, 28, 31–33, 39] 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, 20–23, 30, 37] in establishing refined upper bounds on OUT tailored for more complex degree constraints. Most notably, Khamis et al. [23] proposed the *entropic bound*, which is applicable to any set $\mathsf{DC}$ of degree constraints and is tight in a good sense (see Theorem 5.5 of [37]). 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. [23] introduced the *polymatroid bound* as an alternative, which we represent as $polymat(\mathsf{DC})$ because this bound is fully decided by $\mathsf{DC}$ (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. [23] proposed an algorithm named $\mathsf{PANDA}$ that can evaluate an arbitrary join $\mathcal{Q} \models \mathsf{DC}$ in time $\tilde{O}(polymat(\mathsf{DC}))$.

Interestingly, when $\mathsf{DC}$ is acyclic, the entropic bound is equivalent to the polymatroid bound [30]. In this scenario, Ngo [30] 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 [40]. 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, 24]. Specifically, Kim et al. [24] 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, 24, 40] and the references therein for other results (now superseded) on join sampling.

**Subgraph Listing.** Closely relevant to subgraph sampling is the *subgraph listing* problem. Given a data graph $G = (V, E)$ and a pattern graph $P = (V_P, E_P)$ — both being simple directed graphs — the goal of subgraph listing is to find all the occurrences of $P$ in $G$. Let us define $\rho^*(P)$ — the fractional edge cover number of $P$ — as the fractional edge cover number $\rho^*(P')$ of the corresponding undirected graph $P' = (V_{P'}, E_{P'})$ that is obtained from $P$ by ignoring edge directions; formally, $V_P = V_{P'}$ and $\{u, v\} \in E_{P'}$ if and only if either $(u, v) \in E_P$ or $(v, u) \in E_P$.

When $P$ has a constant size, the data graph $G$ can contain $O(|E|^{\rho^*(P)})$ occurrences of $P$ [4, 6]. Furthermore, this 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 $O(|E|^{\rho^*(P)})$ time. It is well-known that subgraph listing can be converted to a join $\mathcal{Q}$ on binary relations. 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 $join(\mathcal{Q})$ in $O(|E|^{\rho^*(P)})$ extra time. Thus, any $\tilde{O}(AGM(\mathcal{Q}))$-time join algorithm is essentially worst-case optimal for subgraph listing.

However, the tightness of the AGM bound assumes that the vertices in $G$ can have degrees up to $|V| - 1$. Jayaraman et al. [18] studied how to reduce the time of enumerating all pattern occurrences in the more practical scenario where vertices have much lower degrees. For this purpose, they resorted to the polymatroid bound of the aforementioned join $\mathcal{Q}$ derived from subgraph listing. As will be explained in Section 4, this join $\mathcal{Q}$ has a set $\mathsf{DC}$ of degree constraints whose constraint dependency graph $G_{\mathsf{DC}}$ coincides with $P$. Jayaraman et al. [18] gave an algorithm that (after a preprocessing of $O(\text{IN})$ expected time) can list all occurrences of $\mathcal{Q}$ in $G$ within a time complexity that, as we show in Section 4.1, turns out to be $O(polymat(\mathsf{DC}))$. We will delve into their findings further when the specifics become necessary for our discussion.

There is a substantial body of literature on bounding the cost of subgraph listing using parameters different from those already mentioned. These studies typically concentrate on specific patterns (such as paths, cycles, and cliques) or particular graphs (for instance, sparse graphs). We refer interested readers to [1, 7, 8, 11, 14, 17, 19, 26, 29, 38] 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. explained 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 $\mathsf{DC}$ of degree constraints, we will show in Section 3 that it is possible to extract a uniform 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 guarantee is favorable when compared to the recent results of [13, 24] (reviewed in Section 1.2), which are applicable when $\mathsf{DC}$ contains only cardinality constraints and is therefore trivially acyclic. As $\mathit{polymat}(\mathsf{DC})$ 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, 24].

What if $\mathsf{DC}$ is cyclic? An idea, proposed in [30], is to discard enough constraints to make the remaining set $\mathsf{DC}'$ of constraints acyclic (while ensuring $\mathcal{Q} \models \mathsf{DC}'$). Our algorithm can then be applied to draw a sample in $O(\mathit{polymat}(\mathsf{DC}')/\max\{1, \text{OUT}\})$ time. However, $\mathit{polymat}(\mathsf{DC}')$ can potentially be much larger than $\mathit{polymat}(\mathsf{DC})$.

Our next contribution is to prove that the issue does not affect subgraph listing and subgraph sampling. Consider first subgraph listing, defined by a pattern graph $P$ and a data graph $G$ where every vertex has an out-degree at most $\lambda$. As mentioned, this problem can be converted to a join $\mathcal{Q}$ on binary relations, which is associated with a set $\mathsf{DC}$ of degree constraints such that the constraint dependency graph $G_{\mathsf{DC}}$ is exactly $P$ (Section 4.1 will describe the conversion in full). Consequently, whenever $P$ contains a cycle, so does $G_{\mathsf{DC}}$, making $\mathsf{DC}$ cyclic. Nevertheless, we will establish a graph-theoretic theorem — which we name the *De-cycling Theorem* — that guarantees the existence of an acyclic set $\mathsf{DC}' \subset \mathsf{DC}$ ensuring

$$Q \models \mathsf{DC}' \text{ and } \mathit{polymat}(\mathsf{DC}) = \Theta(\mathit{polymat}(\mathsf{DC}')).$$

This "magical" $\mathsf{DC}'$ has an immediate implication: Ngo's join algorithm in [30], when applied to $Q$ and $\mathsf{DC}'$ 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] that has the same guarantees.

The same elegance extends to subgraph sampling: by applying our new join sampling algorithm to $\mathcal{Q}$ and the magical $\mathsf{DC}'$, we can sample an occurrence of $P$ in $G$ using $O(\mathit{polymat}(\mathsf{DC})/\max\{1, \text{OUT}\})$ 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, 24] reviewed in Section 1.2.

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. We will elaborate on the details in Section 5.

## 2 Preliminaries

**Set Functions, Polymatroid Bounds, and Modular Bounds.** Suppose that $\mathcal{V}$ is a finite set. We refer to a function $h : 2^{\mathcal{V}} \to \mathbb{R}_{\geq 0}$ as a *set function over* $\mathcal{V}$, 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 $\mathcal{X} \subset \mathcal{Y} \subseteq \mathcal{V}$;
- *modular* if $h(\mathcal{X}) = \sum_{A \in \mathcal{X}} h(\{A\})$ holds for any $\mathcal{X} \subseteq \mathcal{V}$;
- *submodular* if $h(\mathcal{X} \cup \mathcal{Y}) + h(\mathcal{X} \cap \mathcal{Y}) \leq h(\mathcal{X}) + h(\mathcal{Y})$ holds for any $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{V}$.

Define:

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

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

Consider $\mathcal{C}$ to be a set of triples each having the form $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ where $\mathcal{X} \subset \mathcal{Y} \subseteq \mathcal{V}$ and $N_{\mathcal{Y}|\mathcal{X}} \geq 1$ is an integer. We will refer to $\mathcal{C}$ as a *rule collection* over $\mathcal{V}$. The rule collection instructs us to focus on only those set functions in:

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

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

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

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

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

**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 $\mathsf{DC}$ 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 $\mathsf{DC}$. The set $\mathsf{DC}$ is a rule collection over $\mathcal{V}$. The following lemma was established by Khamis et al. [23]:

**Lemma 2.1** ( [23])**.** *The output size* OUT *of* $\mathcal{Q}$ *is at most* $\mathit{polymat}(\mathsf{DC})$*, i.e., the polymatroid bound of* $\mathsf{DC}$ *defined in* (4)*.*

How about $\mathit{modular}(\mathsf{DC})$, i.e., the modular bound of $\mathcal{V}$? As $\mathsf{M}_{\mathcal{V}} \subseteq \Gamma_{\mathcal{V}}$, we have $\mathit{modular}(\mathsf{DC}) \leq \mathit{polymat}(\mathsf{DC})$ and the inequality can be strict in general. However, an exception arises when $\mathsf{DC}$ is acyclic, as proved in [30]:

**Lemma 2.2** ( [30])**.** *When* $\mathsf{DC}$ *is acyclic,* $\mathit{modular}(\mathsf{DC}) = \mathit{polymat}(\mathsf{DC})$*, 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 $\mathsf{DC}$ is acyclic, $\mathit{modular}(\mathsf{DC})$ 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 polymatroid 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 2.2 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 $h(\mathcal{Y}) - h(\mathcal{X}) = \sum_{A \in \mathcal{Y} \setminus \mathcal{X}} h(A)$ for any $\mathcal{X} \subset \mathcal{Y} \subseteq \mathcal{V}$.

Formally, if we associate each $A \in \mathcal{V}$ with a variable $\nu_A$, then $\max_{h \in \mathsf{M}_{\mathcal{V}} \cap \mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})$ — hence, also $\max_{h \in \Gamma_{\mathcal{V}} \cap \mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})$ for acyclic $\mathsf{DC}$ — is precisely the optimal value of the following LP:

**modular LP** maximize $\sum_{A\in\mathcal{V}} \nu_A$ subject to

$$\sum_{A\in\mathcal{Y}\setminus\mathcal{X}} \nu_A \le \log N_{\mathcal{Y}|\mathcal{X}} \qquad \text{for each } (\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in \mathsf{DC}$$
$$\nu_A \ge 0 \qquad \text{for each } A\in\mathcal{V}$$

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:

**dual modular LP** minimize $\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}} \delta_{\mathcal{Y}|\mathcal{X}}\cdot\log N_{\mathcal{Y}|\mathcal{X}}$ subject to

$$\sum_{\substack{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}\\ \text{such that } A\in\mathcal{Y}\setminus\mathcal{X}}} \delta_{\mathcal{Y}|\mathcal{X}} \ge 1 \quad \text{for each } A\in\mathcal{V}$$
$$\delta_{\mathcal{Y}|\mathcal{X}} \ge 0 \qquad \text{for each } (\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}$$

By the strong duality theorem, the optimal values of the two LPs are equal, meaning that $\max_{h\in\mathsf{M}_{\mathcal{V}}\cap\mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})$ is also the optimal value of the dual modular LP.

Readers may refer to Appendix A for a brief introduction to the basic properties of LP related to duality and its geometric interpretations.

# 3 Join Sampling under Acyclic Degree Dependency

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

**Theorem 3.1.** *For any join $\mathcal{Q}$ consistent with an acyclic set* DC *of degree constraints, we can build in $O(\mathrm{IN})$ expected time a data structure that supports each join sampling operation in $O(\mathit{polymat}(\mathsf{DC})/\max\{1,\mathrm{OUT}\})$ expected time, where* IN *and* OUT *are the input and out sizes of $\mathcal{Q}$, respectively, and $\mathit{polymat}(\mathsf{DC})$ is the polymatroid bound of* DC.

## 3.1 Basic Definitions

Let $\mathcal{G}=(\mathcal{V},\mathcal{E})$ be the schema graph of $\mathcal{Q}$, and $G_{\mathsf{DC}}$ be the constraint dependency graph determined by DC. For each hyperedge $F\in\mathcal{E}$, we denote by $R_F$ the relation in $\mathcal{Q}$ 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 $R_{F(\mathcal{X},\mathcal{Y})}$).

*Example 3.1.* We will illustrate the concepts and algorithms of this section using a running example where $\mathcal{Q}=\{R_{ABC},R_{BCD},R_{ACD},R_{ABD}\}$, with the four relations' content shown in Figure 1. The schema graph of $\mathcal{Q}$ is $\mathcal{G}=(\mathcal{V},\mathcal{E})$, where $\mathcal{V}=\{A,B,C,D\}$ and $\mathcal{E}=\{\{A,B,C\},\{B,C,D\},\{A,C,D\},\{A,B,D\}\}$. Figure 2a shows the degree constraints in DC under columns 2-4, where set symbols are omitted for better clarity (e.g., $\mathcal{Y}=ABC$ should be understood as $\mathcal{Y}=\{A,B,C\}$). For convenient referencing, an ID has been assigned to each constraint (the first column). The last column describes the main guards of all the constraints. Figure 2b gives the constraint dependency graph $G_{\mathsf{DC}}$ determined by DC. □

Set

$$k=|\mathcal{V}|.$$

| $A$ | $B$ | $C$ |
|---|---|---|
| 1 | 3 | 6 |
| 1 | 4 | 2 |
| 1 | 4 | 7 |
| 2 | 1 | 3 |
| 2 | 2 | 1 |
| 2 | 3 | 4 |

$R_{ABC}$

| $A$ | $B$ | $D$ |
|---|---|---|
| 1 | 4 | 2 |
| 1 | 4 | 4 |
| 2 | 2 | 1 |
| 2 | 2 | 2 |
| 2 | 2 | 3 |
| 2 | 2 | 7 |

$R_{ABD}$

| $A$ | $C$ | $D$ |
|---|---|---|
| 1 | 2 | 4 |
| 1 | 7 | 2 |
| 1 | 7 | 4 |
| 2 | 1 | 3 |
| 2 | 5 | 7 |
| 3 | 4 | 6 |

$R_{ACD}$

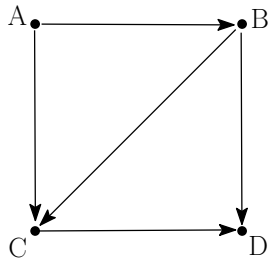
| $B$ | $C$ | $D$ |
|---|---|---|
| 2 | 1 | 3 |
| 2 | 4 | 1 |
| 3 | 5 | 5 |
| 4 | 2 | 4 |
| 4 | 7 | 2 |
| 4 | 7 | 4 |

$R_{BCD}$

Figure 1: Running example: $\mathcal{Q}$ is the join on relations $R_{ABC}, R_{BCD}, R_{ACD}$, and $R_{ABD}$

| constraint ID | $\mathcal{X}$ | $\mathcal{Y}$ | $N_{\mathcal{Y}\|\mathcal{X}}$ | $R_{F(\mathcal{X},\mathcal{Y})}$ |
|---|---|---|---|---|
| 1 | $\emptyset$ | $ABC$ | 6 | $R_{ABC}$ |
| 2 | $\emptyset$ | $ABD$ | 6 | $R_{ABD}$ |
| 3 | $\emptyset$ | $ACD$ | 6 | $R_{ACD}$ |
| 4 | $\emptyset$ | $BCD$ | 6 | $R_{BCD}$ |
| 5 | $\emptyset$ | $AB$ | 2 | $R_{ABD}$ |
| 6 | $A$ | $AB$ | 3 | $R_{ABC}$ |
| 7 | $BC$ | $BCD$ | 2 | $R_{BCD}$ |
| 8 | $AB$ | $ABC$ | 2 | $R_{ABC}$ |

(a) $\mathsf{DC}$




(b) $G_{\mathsf{DC}}$

Figure 2: The degree constraint set $\mathsf{DC}$ and the constraint dependency graph $G_{\mathsf{DC}}$

As $G_{\mathsf{DC}}$ is a DAG (acyclic directed graph), we can order its $k$ vertices (i.e., attributes) into a topological order: $A_1, A_2, ..., A_k$. This means that $G_{\mathsf{DC}}$ cannot have an edge from $A_i$ to $A_j$ where the subscripts satisfy $j < i$; this further implies — by the way $G_{\mathsf{DC}}$ is constructed — that $\mathsf{DC}$ has no constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ satisfying $A_i \in \mathcal{X}$ and $A_j \in \mathcal{Y} \setminus \mathcal{X}$ (because such a constraint would induce an edge from $A_i$ to $A_j$). For any $i \in [k]$, let us define

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

By the aforementioned observations, for each $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)$, the set $\mathcal{X}$ can include only attributes $A_j$ with subscripts $j < i$.

*Example 3.2.* The subsequent discussion will assume the topological order $A, B, C, D$ of the graph $G_{\mathsf{DC}}$ in Figure 2b. Accordingly, we can derive from (6):

$$\begin{aligned}
\mathsf{DC}(A_1) = \mathsf{DC}(A) &= \text{the set of constraints with IDs 1, 2, 3, and 5} \\
\mathsf{DC}(A_2) = \mathsf{DC}(B) &= \text{the set of constraints with IDs 1, 2, 4, 5, and 6} \\
\mathsf{DC}(A_3) = \mathsf{DC}(C) &= \text{the set of constraints with IDs 1, 3, 4, and 8} \\
\mathsf{DC}(A_4) = \mathsf{DC}(D) &= \text{the set of constraints with IDs 2, 3, 4, and 7.}
\end{aligned}$$

Constraint IDs are as shown in Figure 2a. □

For convenience, let us introduce for each $i \in [0, k]$:

$$\mathcal{V}_i = \begin{cases} \emptyset & \text{if } i = 0 \\ \{A_1, A_2, ..., A_i\} & \text{if } i \geq 1 \end{cases} \tag{7}$$

We now define an important concept called "relative degree". For this purpose, fix

- an arbitrary $i \in [k]$;
- an arbitrary tuple $\boldsymbol{w}$ over $\mathcal{V}_{i-1}$ — note: if $i = 1$, then $\mathcal{V}_{i-1} = \emptyset$ and $\boldsymbol{w}$ is the null tuple;
- an arbitrary constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)$ — note: recall from our earlier discussion that $\mathcal{X}$ can contain only attributes from $\mathcal{V}_i$, because of which $\boldsymbol{w}$ must have set a value for every attribute in $\mathcal{X}$ (but $\boldsymbol{w}$ has not set any value for $A_i$ as $A_i \in \mathcal{Y} \setminus \mathcal{X}$);
- an arbitrary value $v \in \mathbf{dom}$.

Then, the *relative degree* of $(\boldsymbol{w}, v)$ in $R_{F(\mathcal{X},\mathcal{Y})}$ is:

$$\mathit{reldeg}_{\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, v) \quad = \quad \begin{cases} 0 & \text{if } R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w} = \emptyset \\ \dfrac{\left|\sigma_{A_i=v}(\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}))\right|}{\left|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})\right|} & \text{otherwise} \end{cases} \tag{8}$$

where $\sigma$ and $\ltimes$ are the standard selection and semi-join operators in relational algebra, respectively. To understand the intuition behind (8), consider $(\boldsymbol{w}, v)$ as a tuple over $\mathcal{V}_i$, i.e., "extending" $\boldsymbol{w}$ by setting $A_i = v$. Then, the numerator (in the second branch) of (8) is the size of $\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes (\boldsymbol{w}, v))$. It is thus clear that (8) gives the fraction that $\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes (\boldsymbol{w}, v))$ accounts for in $\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})$.

*Example 3.3.* As mentioned in Example 3.2, the topological order for our running example is $A_1 = A, A_2 = B, A_3 = C$, and $A_4 = D$. Consider $i = 2$ and $\boldsymbol{w}$ as a tuple over $\{A_1\}$ with $\boldsymbol{w}(A) = 2$. Furthermore, choose from $\mathsf{DC}(A_2) = \mathsf{DC}(B)$ a constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) = (A, AB, 3)$ (with ID 6), whose main guard is $R_{F(\mathcal{X},\mathcal{Y})} = R_{ABC}$ (see Figure 2a). Finally, fix value $v = 2$.

From Figure 1, we can see that $R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}$ has three tuples $\{(2,1,3), (2,2,1), (2,3,4)\}$ (over schema $ABC$) and thus $\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}) = \{(2,1), (2,2), (2,3)\}$ (over schema $AB$). Among the tuples in $\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})$, only $(2,2)$ satisfies $B = 2$. Thus, $|\sigma_{A_i=v}(\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}))| = 1$. By (8), the relative degree of $(\boldsymbol{w}, v)$ in $R_{ABC}$ is $\mathit{reldeg}_{\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, v) = 1/3$. □

Next, we extend "relative degree" to another concept called "maximum relative degree". For this purpose, fix

- an arbitrary $i \in [k]$;
- an arbitrary tuple $\boldsymbol{w}$ over $\mathcal{V}_{i-1}$;
- an arbitrary value $v \in \mathbf{dom}$.

Then, the *maximum relative degree* of $(\boldsymbol{w}, v)$ is:

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

Conceptually, for every degree constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)$, we calculate the relative degree of $(\boldsymbol{w}, v)$ in $R_{F(\mathcal{X},\mathcal{Y})}$, after which $\mathit{reldeg}^*(\boldsymbol{w}, v)$ is the maximum of all those relative degrees. The following "companion" definition aims to capture which degree constraint attains the maximum:

$$\mathit{constraint}^*(\boldsymbol{w}, v) \quad = \quad \operatorname*{arg\,max}_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(A_i)} \mathit{reldeg}_{\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, v). \tag{10}$$

*Example 3.4.* Let us first consider $i = 1$, $\boldsymbol{w}$ as the null tuple, and $v = 2$. As noted in Example 3.2, the set $\mathsf{DC}(A_i) = \mathsf{DC}(A)$ has four constraints: $(\emptyset, ABC, 6), (\emptyset, ABD, 6), (\emptyset, ACD, 6)$, and $(\emptyset, AB, 2)$. As shown in Figure 2a, their main guards are $R_{ABC}$, $R_{ABD}$, $R_{ACD}$, and $R_{ABD}$, respectively. We have from (8):

$$\begin{aligned}
\mathit{reldeg}_{\emptyset,ABC}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2}(\Pi_{ABC}(R_{ABC}))|}{|\Pi_{ABC}(R_{ABC})|} = \frac{3}{6} = \frac{1}{2} \\
\mathit{reldeg}_{\emptyset,ABD}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2}(\Pi_{ABD}(R_{ABD}))|}{|\Pi_{ABD}(R_{ABD})|} = \frac{4}{6} = \frac{2}{3} \\
\mathit{reldeg}_{\emptyset,ACD}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2}(\Pi_{ACD}(R_{ACD}))|}{|\Pi_{ACD}(R_{ACD})|} = \frac{2}{6} = \frac{1}{3} \\
\mathit{reldeg}_{\emptyset,AB}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2}(\Pi_{AB}(R_{ABD}))|}{|\Pi_{AB}(R_{ABD})|} = \frac{1}{2}.
\end{aligned}$$

Hence, $\mathit{reldeg}^*(\boldsymbol{w}, v) = 2/3$ and $\mathit{constraint}^*(\boldsymbol{w}, v) = (\emptyset, ABD, 6)$.

As another example, consider $i = 2$, $\boldsymbol{w}$ as a tuple over $\{A\}$ with $\boldsymbol{w}(A) = 2$, and $v = 2$. The set $\mathsf{DC}(A_i) = \mathsf{DC}(B)$ has five constraints: $(\emptyset, ABC, 6), (\emptyset, ABD, 6), (\emptyset, BCD, 6), (\emptyset, AB, 2)$, and $(A, AB, 3)$, whose main guards are $R_{ABC}$, $R_{ABD}$, $R_{BCD}$, $R_{ABD}$, and $R_{ABC}$, respectively. We have:

$$\begin{aligned}
\mathit{reldeg}_{\emptyset,ABC}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2,B=2}(\Pi_{ABC}(R_{ABC}))|}{|\sigma_{A=2}(\Pi_{ABC}(R_{ABC}))|} = \frac{1}{3} \\
\mathit{reldeg}_{\emptyset,ABD}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2,B=2}(\Pi_{ABD}(R_{ABD}))|}{|\sigma_{A=2}(\Pi_{ABD}(R_{ABD}))|} = \frac{4}{4} = 1 \\
\mathit{reldeg}_{\emptyset,BCD}(\boldsymbol{w}, v) &= \frac{|\sigma_{B=2}(\Pi_{BCD}(R_{BCD}))|}{|\Pi_{BCD}(R_{BCD})|} = \frac{2}{6} = \frac{1}{3} \\
\mathit{reldeg}_{\emptyset,AB}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2,B=2}(\Pi_{AB}(R_{ABD}))|}{|\sigma_{B=2}(\Pi_{AB}(R_{ABD}))|} = \frac{1}{1} = 1 \\
\mathit{reldeg}_{A,AB}(\boldsymbol{w}, v) &= \frac{|\sigma_{A=2,B=2}(\Pi_{AB}(R_{ABC}))|}{|\sigma_{B=2}(\Pi_{AB}(R_{ABC}))|} = \frac{1}{3}.
\end{aligned}$$

Hence, $\mathit{reldeg}^*(\boldsymbol{w}, v) = 1$ and $\mathit{constraint}^*(\boldsymbol{w}, v) = (\emptyset, ABD, 6)$ (alternatively, one may also set $\mathit{constraint}^*(\boldsymbol{w}, v) = (\emptyset, AB, 2)$). □

### 3.2 Global and Local Polymatroid Bounds

Henceforth, we will fix an arbitrary optimal solution

$$\{\delta^*_{\mathcal{Y}|\mathcal{X}} \mid (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}\} \tag{11}$$

to the dual modular LP in Section 2. Thus:

$$\begin{aligned}
\prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}} N_{\mathcal{Y}|\mathcal{X}}^{\delta^*_{\mathcal{Y}|\mathcal{X}}} &= \exp_2\Big(\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}} \delta^*_{\mathcal{Y}|\mathcal{X}} \cdot \log N_{\mathcal{Y}|\mathcal{X}}\Big) = \exp_2\Big(\max_{h\in\mathsf{M}_{\mathcal{V}}\cap\mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})\Big) \\
\text{(by (5))} &= \mathit{modular}(\mathsf{DC}) \\
\text{(by Lemma 2.2)} &= \mathit{polymat}(\mathsf{DC}).
\end{aligned} \tag{12}$$

*Example 3.5.* In Figure 2a, we have assigned an ID to each degree constraint in $\mathsf{DC}$. For each $i \in [1, 8]$, introduce $\delta_i$ as an alias for the variable $\delta_{\mathcal{Y}|\mathcal{X}}$ of the constraint with ID $i$. Furthermore,

define $N_i$ as the value in the column $N_{\mathcal{Y}|\mathcal{X}}$ (of Figure 2a) corresponding to the constraint with ID $i$ (e.g., $N_1 = 6, N_5 = 2$, and $N_6 = 3$). Then, the dual modular LP for our running example is:

$$\text{minimize } \sum_{i=1}^{8} \delta_i \cdot \log N_i \text{ subject to}$$

$$\begin{array}{lll}
\delta_i \geq 0 & \text{for each } i \in [8] & \\
\delta_1 + \delta_2 + \delta_3 + \delta_5 \geq 1 & \text{this corresponds to} & \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in \mathsf{DC}(A)} \delta_{\mathcal{Y}|\mathcal{X}} \geq 1 \\
\delta_1 + \delta_2 + \delta_4 + \delta_5 + \delta_6 \geq 1 & \text{this corresponds to} & \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in \mathsf{DC}(B)} \delta_{\mathcal{Y}|\mathcal{X}} \geq 1 \\
\delta_1 + \delta_2 + \delta_4 + \delta_8 \geq 1 & \text{this corresponds to} & \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in \mathsf{DC}(C)} \delta_{\mathcal{Y}|\mathcal{X}} \geq 1 \\
\delta_2 + \delta_3 + \delta_4 + \delta_7 \geq 1 & \text{this corresponds to} & \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in \mathsf{DC}(D)} \delta_{\mathcal{Y}|\mathcal{X}} \geq 1
\end{array}$$

The following values make an optimal solution to the above LP:

$$\begin{aligned}
\delta_{ABC|\emptyset} = \delta_1 &= 0 \\
\delta_{ABD|\emptyset} = \delta_2 &= 0 \\
\delta_{ACD|\emptyset} = \delta_3 &= 0 \\
\delta_{BCD|\emptyset} = \delta_4 &= 0 \\
\delta_{AB|\emptyset} = \delta_5 &= 1 \\
\delta_{AB|A} = \delta_6 &= 0 \\
\delta_{BCD|BC} = \delta_7 &= 1 \\
\delta_{ABC|AB} = \delta_8 &= 1
\end{aligned}$$

Define $\delta^*_{ABC|\emptyset} = \delta^*_{ABD|\emptyset} = \delta^*_{ACD|\emptyset} = \delta^*_{BCD|\emptyset} = \delta^*_{AB|A} = 0$ and $\delta^*_{AB|\emptyset} = \delta^*_{BCD|BC} = \delta^*_{ABC|AB} = 1$. We thus have:

$$\begin{aligned}
\mathit{polymat}(\mathsf{DC}) &= \exp_2\Big(\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in \mathsf{DC}} \delta^*_{\mathcal{Y}|\mathcal{X}} \cdot \log N_{\mathcal{Y}|\mathcal{X}}\Big) \\
&= \exp_2\Big(\log N_{AB|\emptyset} + \log N_{BCD|BC} + \log N_{ABC|AB}\Big) \\
&= N_{AB|\emptyset} \cdot N_{BCD|BC} \cdot N_{ABC|AB} \\
&= 2 \cdot 2 \cdot 2 = 8.
\end{aligned}$$

The values of $N_{AB|\emptyset}$, $N_{BCD|BC}$, and $N_{ABC|AB}$ are shown in Figure 2a. □

The polymatroid bound in (12) is "global" because it is an upper bound on the size of the whole result $\mathit{join}(\mathcal{Q})$. Next, we will introduce its "local" counterpart. For this purpose, fix

- an arbitrary integer $i \in [0, k]$, and
- an arbitrary tuple $\boldsymbol{w}$ over $\mathcal{V}_i$ (see (7)).

Define:

$$B(\boldsymbol{w}) = \exp_2\Big(\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in \mathsf{DC}} \delta^*_{\mathcal{Y}|\mathcal{X}} \cdot \log \mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})\Big). \tag{13}$$

The lemma below explains why this is a local polymatroid bound. In other words, if we filter the join result $join(\mathcal{Q})$ by discarding all those tuples inconsistent with the values of $\boldsymbol{w}$, the number of remaining tuples in $join(\mathcal{Q})$ is at most $B(\boldsymbol{w})$.

**Lemma 3.2.** $|join(\mathcal{Q}) \bowtie \boldsymbol{w}| \leq B(\boldsymbol{w})$.

*Proof.* Define a join

$$\mathcal{Q}(\boldsymbol{w}) = \{R \ltimes \boldsymbol{w} \mid R \in \mathcal{Q}\}.$$

It is clear that $join(\mathcal{Q}) \bowtie \boldsymbol{w} = join(\mathcal{Q}(\boldsymbol{w}))$. The subsequent proof will show $|join(\mathcal{Q}(\boldsymbol{w}))| \leq B(\boldsymbol{w})$.

Construct a set of constraints $\mathsf{DC}(\boldsymbol{w})$ as follows. Initially, set $\mathsf{DC}(\boldsymbol{w}) = \emptyset$. For each constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$, add $(\mathcal{X}, \mathcal{Y}, N'_{\mathcal{Y}|\mathcal{X}})$ to $\mathsf{DC}(\boldsymbol{w})$, where $N'_{\mathcal{Y}|\mathcal{X}} = deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w})$. As far as $\mathcal{Q}(\boldsymbol{w})$ is concerned, this constraint is guarded by the relation $R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}$. Hence, $\mathcal{Q}(\boldsymbol{w})$ is compatible with $\mathsf{DC}(\boldsymbol{w})$.

The constraint dependency graph of $\mathsf{DC}(\boldsymbol{w})$ is identical to that of $\mathsf{DC}$. Thus, $\mathsf{DC}(\boldsymbol{w})$ is acyclic; and Lemma 2.1 tells us that $|join(Q(\boldsymbol{w}))| \leq polymat(\mathsf{DC}(\boldsymbol{w}))$. Next, we will prove:

$$polymat(\mathsf{DC}(\boldsymbol{w})) \leq B(\boldsymbol{w}). \tag{14}$$

It will then follow that $|join(\mathcal{Q}(\boldsymbol{w}))| \leq polymat(\mathsf{DC}(\boldsymbol{w})) \leq B(\boldsymbol{w})$.

By Lemma 2.2, we have $polymat(\mathsf{DC}(\boldsymbol{w})) = modular(\mathsf{DC}(\boldsymbol{w}))$, while as discussed in Section 2 $\log(modular(\mathsf{DC}(\boldsymbol{w})))$ is the value obtained by the dual modular LP:

$$\begin{aligned}
\text{minimize} \sum_{(\mathcal{X},\mathcal{Y},N'_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(\boldsymbol{w})} & \delta_{\mathcal{Y}|\mathcal{X}} \cdot \log N'_{\mathcal{Y}|\mathcal{X}} \text{ subject to} \\
\sum_{\substack{(\mathcal{X},\mathcal{Y},N'_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(\boldsymbol{w}) \\ \text{such that } A\in\mathcal{Y}\setminus\mathcal{X}}} & \delta_{\mathcal{Y}|\mathcal{X}} \geq 1 && \text{for each } A \in \mathcal{V} \\
& \delta_{\mathcal{Y}|\mathcal{X}} \geq 0 && \text{for each } (\mathcal{X}, \mathcal{Y}, N'_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}(\boldsymbol{w})
\end{aligned}$$

Consider the set $\{\delta^*_{\mathcal{Y}|\mathcal{X}} \mid (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}\}$ in (11), namely, an optimal solution to the dual modular LP defined by $\mathsf{DC}$. This is also a feasible solution to the above LP defined by $\mathsf{DC}(\boldsymbol{w})$. Hence, the value returned by this LP cannot exceed

$$\begin{aligned}
\sum_{(\mathcal{X},\mathcal{Y},N'_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(\boldsymbol{w})} \delta^*_{\mathcal{Y}|\mathcal{X}} \cdot \log N'_{\mathcal{Y}|\mathcal{X}} &= \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}} \delta^*_{\mathcal{Y}|\mathcal{X}} \cdot \log deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}) \\
&= \log(B(\boldsymbol{w})).
\end{aligned}$$

We now have $modular(\mathsf{DC}(\boldsymbol{w})) \leq B(\boldsymbol{w})$, which then gives (14) (because $polymat(\mathsf{DC}(\boldsymbol{w})) = modular(\mathsf{DC}(\boldsymbol{w}))$, as mentioned). □

*Example 3.6.* Let us first consider $i = 1$ and $\boldsymbol{w}$ as the tuple over $\mathcal{V}_1 = \{A\}$ with $\boldsymbol{w}(A) = 2$. Utilizing the optimal LP solution in Example 3.5 and also the main-guard relations shown in Figure 2a, we have

$$\begin{aligned}
B(\boldsymbol{w}) = \exp_2 \Big( & \delta^*_{AB|\emptyset} \cdot \log deg_{AB|\emptyset}(R_{ABD} \ltimes \boldsymbol{w}) + \delta^*_{BCD|BC} \cdot \log deg_{BCD|BC}(R_{BCD} \ltimes \boldsymbol{w}) \\
& \delta^*_{ABC|AB} \cdot \log deg_{ABC|AB}(R_{ABC} \ltimes \boldsymbol{w}) \Big)
\end{aligned}$$

$$= deg_{AB|\emptyset}(R_{ABD} \ltimes \boldsymbol{w}) \cdot deg_{BCD|BC}(R_{BCD} \ltimes \boldsymbol{w}) \cdot deg_{ABC|AB}(R_{ABC} \ltimes \boldsymbol{w})$$
$$= 1 \cdot 2 \cdot 1 = 2.$$

Lemma 3.2 assures us that at most 2 tuples $\boldsymbol{u} \in join(\mathcal{Q})$ can satisfy $\boldsymbol{u}(A) = 2$.

As another example, let us consider $i = 2$ and $\boldsymbol{w}$ as the tuple over $\mathcal{V}_2 = \{A, B\}$ with $\boldsymbol{w}(A) = \boldsymbol{w}(B) = 2$. This time, we have:

$$\begin{aligned} B(\boldsymbol{w}) &= \exp_2\Big(\delta^*_{AB|\emptyset} \cdot \log deg_{AB|\emptyset}(R_{ABD} \ltimes \boldsymbol{w}) + \delta^*_{BCD|BC} \cdot \log deg_{BCD|BC}(R_{BCD} \ltimes \boldsymbol{w}) \\ &\quad + \delta^*_{ABC|AB} \cdot \log deg_{ABC|AB}(R_{ABC} \ltimes \boldsymbol{w})\Big) \\ &= deg_{AB|\emptyset}(R_{ABD} \ltimes \boldsymbol{w}) \cdot deg_{BCD|BC}(R_{BCD} \ltimes \boldsymbol{w}) \cdot deg_{ABC|AB}(R_{ABC} \ltimes \boldsymbol{w}) \\ &= 1 \cdot 1 \cdot 1 = 1. \end{aligned}$$

Lemma 3.2 assures us that at most one tuple $\boldsymbol{u} \in join(\mathcal{Q})$ can satisfy $\boldsymbol{u}(A) = 2$ and $\boldsymbol{u}(B) = 2$. □

The following simple observations will be useful later:

- If $i = 0$, then $\boldsymbol{w}$ is the null tuple and $B(\boldsymbol{w}) = polymat(\mathsf{DC})$.
- If $i = k$ and $\boldsymbol{w} \in join(\mathcal{Q})$, then $B(\boldsymbol{w}) = 1$.

### 3.3 The Core: ADC-Sample

Figure 3 presents ADC-sample (where ADC stands for acyclic dependence constraints), which is the core of our sampling method. At a high level, ADC-sample processes one attribute at a time according to the topological order $A_1, A_2, ..., A_k$. The for-loop in Lines 2-8 finds a value $v_i$ for attribute $A_i$ ($i \in [k]$). The algorithm may fail to return anything, but when it *succeeds* (i.e., returning at Line 9), the values $v_1, v_2, ..., v_k$ will form a uniformly random tuple from $join(\mathcal{Q})$.

Next, we explain the for-loop. Suppose that, in the first $i-1$ iterations of the for-loop, the algorithm has already found the values $v_1, ..., v_{i-1}$ for $A_1, ..., A_{i-1}$, respectively. These values are stored in tuple $\boldsymbol{w}_{i-1}$ (i.e., $\boldsymbol{w}_{i-1}(A_j) = v_j$ for all $j \in [i-1]$). The $i$-th iteration is designed to achieve the following purpose: for any tuple $\boldsymbol{u} \in join(\mathcal{Q})$ with $\boldsymbol{u}[\{A_1, ..., A_{i-1}\}] = \boldsymbol{w}_{i-1}$, sample the value $\boldsymbol{u}(A_i)$ for attribute $A_i$ with a probability proportional to $B(\boldsymbol{u}[\{A_1, ..., A_{i-1}, A_i\}])/B(\boldsymbol{w}_{i-1})$.

Let us now delve into the details. 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) for the definition of $\mathsf{DC}(A_i)$). Conceptually, 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 $v_i$ (note: $A_i \in \mathcal{Y}^\circ - \mathcal{X}^\circ$). Physically, we do not compute $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_{i-1})$ during the sampling process; instead, with proper preprocessing (discussed later), we can acquire the value $v_i$ 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 $constraint^*(\boldsymbol{w}_{i-1}, v_i)$ (whose definition is in (10)). 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 $v_j$ on attribute $A_j$ for each $j \in [i]$ (Line 7). Line 8 allows us to pass with probability

$$p_{\text{pass}}(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i) = \frac{B(\boldsymbol{w}_i)}{B(\boldsymbol{w}_{i-1})} \cdot \frac{1}{reldeg^*(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i(A_i))} \tag{15}$$

ADC-sample

0. $A_1, A_2, ..., A_k \leftarrow$ a topological order of $G_{\mathsf{DC}}$
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. $v_i \leftarrow \boldsymbol{u}^\circ(A_i)$
6. **if** $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) \neq \mathit{constraint}^*(\boldsymbol{w}_{i-1}, v_i)$ **then** declare **failure**
7. $\boldsymbol{w}_i \leftarrow$ the tuple over $\mathcal{V}_i$ formed by extending $\boldsymbol{w}_{i-1}$ with $A_i = v_i$
8. declare **failure** with probability $1 - p_{\text{pass}}(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i)$, where $p_{\text{pass}}(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i)$ is given in (15)
9. **if** $\boldsymbol{w}_k[F] \in R_F$ for $\forall F \in \mathcal{E}$ **then return** $\boldsymbol{w}_k$ /* that is, $\boldsymbol{w}_k \in \mathit{join}(\mathcal{Q})$ */
10. **else** declare **failure**

Figure 3: The ADC-sample algorithm

or otherwise terminate the algorithm by declaring failure. As proved later, $p_{\text{pass}}(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i)$ cannot exceed 1 (Lemma 3.3); moreover, this value can be computed in $O(1)$ time (Appendix B). The overall execution time of ADC-sample is constant.

*Example 3.7.* We will illustrate ADC-sample using our running example in Figures 1a and 2 according to the topological order of attributes $A_1 = A, A_2 = B, A_3 = C$, and $A_4 = D$. Recall that we have obtained an optimal solution to the dual modular LP in Example 3.5.

At the outset, $\boldsymbol{w}_0 =$ null and $i = 1$. Suppose that, from $\mathsf{DC}(A_1) = \mathsf{DC}(A)$ (which can be found in Example 3.2), Line 3 randomly chooses $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) = (\emptyset, ABD, 6)$. Here, $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_0)$ is simply the entire $R_{ABD}$. Thus, Line 4 randomly picks a tuple $\boldsymbol{u}^\circ$ from $R_{ABD}$; for our discussion, let it be $(2, 2, 1)$, which gives $v_1 = 2$ at Line 5. Line 6 would output "failure" if $\mathit{constraint}^*(\boldsymbol{w}_0, 2)$ was not $(\emptyset, ABD, 6)$. However, as shown in Example 3.4, $\mathit{constraint}^*(\boldsymbol{w}_0, 2)$ is indeed $(\emptyset, ABD, 6)$. Hence, ADC-sample creates tuple $\boldsymbol{w_1}$ with $\boldsymbol{w_1}(A) = 2$. Then, the algorithm calculates

$$\begin{aligned} p_{\text{pass}}(\boldsymbol{w}_0, \boldsymbol{w}_1) &= \frac{B(\boldsymbol{w}_1)}{B(\boldsymbol{w}_0)} \cdot \frac{1}{\mathit{reldeg}^*(\boldsymbol{w}_0, 2)} = \frac{B(\boldsymbol{w}_1)}{\mathit{polymat}(\mathsf{DC})} \cdot \frac{1}{\mathit{reldeg}^*(\boldsymbol{w}_0, 2)} \\ &= \frac{2}{8} \cdot \frac{1}{2/3} = \frac{3}{8} \end{aligned}$$

where the derivation of $B(\boldsymbol{w}_1)$, $\mathit{polymat}(\mathsf{DC})$, and $\mathit{reldeg}^*(\boldsymbol{w}_0, 2)$ can be found in Examples 3.6, 3.5, and 3.4, respectively. Accordingly, Line 8 generates a random number $x$ from 0 to 1 and terminates with "failure" if $x > 3/8$. Here, let us assume $x \leq 3/8$ so the execution continues.

We now return to Line 2 with $i = 2$. Suppose that, from $\mathsf{DC}(A_2) = \mathsf{DC}(B)$, Line 3 again picks $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) = (\emptyset, ABD, 6)$. Here, $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_1)$ includes the following tuples from $R_{ABD}$: $(2, 2, 1), (2, 2, 2), (2, 2, 3)$, and $(2, 2, 7)$. From them, let us assume that Line 4 randomly picks $\boldsymbol{u}^\circ = (2, 2, 3)$, yielding $v_2 = 2$ at Line 5. Example 3.4 has shown that $\mathit{constraint}^*(\boldsymbol{w}_1, 2)$ is exactly $(\emptyset, ABD, 6)$, allowing the execution to get past Line 6. Line 7 now creates $\boldsymbol{w}_2$, which is a tuple over $\{A, B\}$ with $\boldsymbol{w}_2(A) = 2$ and $\boldsymbol{w}_2(B) = 2$. Next, the algorithm calculates

$$p_{\text{pass}}(\boldsymbol{w}_1, \boldsymbol{w}_2) = \frac{B(\boldsymbol{w}_2)}{B(\boldsymbol{w}_1)} \cdot \frac{1}{\mathit{reldeg}^*(\boldsymbol{w}_1, 2)} = \frac{1}{2} \cdot \frac{1}{1} = \frac{1}{2}$$

where the derivation of $B(\boldsymbol{w}_1)$ and $B(\boldsymbol{w}_2)$ can be found in Example 3.6, and that of $\mathit{reldeg}^*(\boldsymbol{w}_1, 2)$ in Example 3.4. Hence, Line 8 gets past with probability 1/2. The rest execution is similar and omitted. □

In Appendix B, we will explain how to preprocess the relations of $\mathcal{Q}$ in $O(\mathrm{IN})$ expected time to ensure that ADC-sample runs in $O(1)$ time. We now proceed to analyze ADC-sample, starting with a lemma suggesting that the value in (15) serves as a legal probability value.

**Lemma 3.3.** *For every $i \in [k]$, it holds that $p_{\mathrm{pass}}(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i) \leq 1$.*

*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 $G_{\mathsf{DC}}$, 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 $|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)|$.

Setting $v = \boldsymbol{w}_i[A_i]$, we can derive:

$$\begin{aligned}
\frac{\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)}{\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})} &= \frac{|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)|}{|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})|} \\
&= \frac{\left|\sigma_{A_i=v}(\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1}))\right|}{\left|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})\right|} \\
&= \mathit{reldeg}_{\mathcal{X},\mathcal{Y}}(\boldsymbol{w}_{i-1}, v) \\
&\leq \mathit{reldeg}^*(\boldsymbol{w}_{i-1}, v). \qquad (16)
\end{aligned}$$

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

$$\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i) \leq \mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1}) \qquad (17)$$

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

$$\begin{aligned}
p_{\mathrm{pass}}(\boldsymbol{w}_{i-1}, \boldsymbol{w}_i) &= \frac{1}{\mathit{reldeg}^*(\boldsymbol{w}_{i-1}, v)} \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}} \left(\frac{\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)}{\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})}\right)^{\delta^*_{\mathcal{Y}|\mathcal{X}}} \\
(\text{by (17)}) \quad &\leq \frac{1}{\mathit{reldeg}^*(\boldsymbol{w}_{i-1}, v)} \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(A_i)} \left(\frac{\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)}{\mathit{deg}_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})}\right)^{\delta^*_{\mathcal{Y}|\mathcal{X}}} \\
(\text{by (16)}) \quad &\leq \frac{1}{\mathit{reldeg}^*(\boldsymbol{w}_{i-1}, v)} \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(A_i)} \mathit{reldeg}^*(\boldsymbol{w}_{i-1}, v)^{\delta^*_{\mathcal{Y}|\mathcal{X}}} \\
&= \mathit{reldeg}^*(\boldsymbol{w}_{i-1}, v)^{\left(\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}(A_i)} \delta^*_{\mathcal{Y}|\mathcal{X}}\right)-1} \leq 1.
\end{aligned}$$

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 (see the discussion in Section 3.2). □

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

- event $\mathbf{E1}(i)$: $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) = \mathit{constraint}^*(\boldsymbol{w}_{i-1}, \boldsymbol{u}(A_i))$ 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 $\boldsymbol{u}$ can be derived as follows.

$$\begin{aligned}
\mathbf{Pr}[\boldsymbol{u} \text{ returned}] &= \prod_{i=1}^{k} \mathbf{Pr}[v_i = \boldsymbol{u}(A_i), \mathbf{E1}(i), \mathbf{E2}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]] \\
&\quad \text{(if } i = 1\text{, then } \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}] \text{ becomes } \boldsymbol{w}_0 = \boldsymbol{u}[\emptyset]\text{, which is vacuously true)} \\
&= \prod_{i=1}^{k} \Big( \mathbf{Pr}[v_i = \boldsymbol{u}(A_i), \mathbf{E1}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]] \cdot \\
&\qquad \mathbf{Pr}[\mathbf{E2}(i) \mid \mathbf{E1}(i), v_i = \boldsymbol{u}(A_i), \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]] \Big). \qquad (18)
\end{aligned}$$

Observe

$$\begin{aligned}
& \mathbf{Pr}[v_i = \boldsymbol{u}(A_i), \mathbf{E1}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]] \\
= \; & \mathbf{Pr}[\mathbf{E1}(i) \mid \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]] \cdot \mathbf{Pr}[v_i = \boldsymbol{u}(A_i) \mid \mathbf{E1}(i), \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]] \\
= \; & \frac{1}{|\mathsf{DC}(A_i)|} \cdot \frac{\left|\sigma_{A_i = \boldsymbol{u}(A_i)}(\Pi_{\mathcal{Y}}(R_{F(\mathcal{X}^\circ, \mathcal{Y}^\circ)} \ltimes \boldsymbol{u}[\mathcal{V}_{i-1}]))\right|}{\left|\Pi_{\mathcal{Y}}(R_{F(\mathcal{X}^\circ, \mathcal{Y}^\circ)} \ltimes \boldsymbol{u}[\mathcal{V}_{i-1}])\right|} \\
& \text{(note: } (\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) = \mathit{constraint}^*(\boldsymbol{u}[\mathcal{V}_{i-1}], \boldsymbol{u}(A_i)) \text{, due to } \mathbf{E1}(i) \text{ and } \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]]\text{)} \\
= \; & \frac{1}{|\mathsf{DC}(A_i)|} \cdot \mathit{reldeg}_{\mathcal{X}^\circ, \mathcal{Y}^\circ}(\boldsymbol{u}[\mathcal{V}_{i-1}], \boldsymbol{u}(A_i)) \\
= \; & \frac{1}{|\mathsf{DC}(A_i)|} \cdot \mathit{reldeg}^*(\boldsymbol{u}[\mathcal{V}_{i-1}], \boldsymbol{u}(A_i)). \qquad (19)
\end{aligned}$$

On the other hand:

$$\begin{aligned}
\mathbf{Pr}[\mathbf{E2}(i) \mid \mathbf{E1}(i), v_i = \boldsymbol{u}(A_i), \boldsymbol{w}_{i-1} = \boldsymbol{u}[\mathcal{V}_{i-1}]] &= p_{\text{pass}}(\boldsymbol{u}[\mathcal{V}_{i-1}], \boldsymbol{u}[\mathcal{V}_i]) \\
\text{(by (15))} &= \frac{B(\boldsymbol{u}[\mathcal{V}_i])}{B(\boldsymbol{u}[\mathcal{V}_{i-1}])} \cdot \frac{1}{\mathit{reldeg}^*(\boldsymbol{u}[\mathcal{V}_{i-1}], \boldsymbol{u}(A_i))}. \qquad (20)
\end{aligned}$$

Plugging (19) and (20) into (18) yields

$$\begin{aligned}
\mathbf{Pr}[\boldsymbol{u} \text{ returned}] &= \prod_{i=1}^{k} \frac{B(\boldsymbol{u}[\mathcal{V}_i])}{B(\boldsymbol{u}[\mathcal{V}_{i-1}])} \cdot \frac{1}{|\mathsf{DC}(A_i)|} = \frac{B(\boldsymbol{u}[\mathcal{V}_k])}{B(\boldsymbol{u}[\mathcal{V}_0])} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|} \\
&= \frac{1}{B(\text{null})} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|} \\
&= \frac{1}{\mathit{polymat}(\mathsf{DC})} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|}.
\end{aligned}$$

As the above is identical for every $\boldsymbol{u} \in \mathit{join}(\mathcal{Q})$, 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

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

We thus expect to find a sample from $\mathit{join}(\mathcal{Q})$ by repeating ADC-sample $O(\mathit{polymat}(\mathsf{DC})/\text{OUT})$ times.

### 3.4 Completing the Proof of Theorem 3.1

Recall that, in join sampling, we are supposed to return a uniform sample of $join(\mathcal{Q})$ or declare $join(\mathcal{Q}) = \emptyset$. The ADC-sample algorithm alone does not fulfill the purpose because it will never succeed if OUT = 0. This issue can be remedied by executing two *threads* concurrently:

- The first thread repeatedly invokes ADC-sample until it manages to return a sample.
- The other thread runs Ngo's algorithm in [30] to compute $join(\mathcal{Q})$ in full, after which we can declare $join(\mathcal{Q}) \neq \emptyset$ or sample from $join(\mathcal{Q})$ in constant time.

As soon as one thread finishes, we manually terminate the other one.

The above two-thread strategy guarantees that a join sampling operation can be completed in $O(polymat(\mathsf{DC})/\max\{1, \text{OUT}\})$ expected time. To see why, consider first the scenario where OUT $\geq 1$. In this case, we expect to find a sample with $O(polymat(\mathsf{DC})/\text{OUT})$ repeats of ADC-sample. Hence, the first thread finishes in $O(polymat(\mathsf{DC})/\text{OUT})$ expected time. On the other hand, if OUT = 0, the second thread will finish in $O(polymat(\mathsf{DC}))$ time. This concludes the proof of Theorem 3.1.

**Remarks.** When DC has only cardinality constraints (is thus "trivially" acyclic), ADC-sample simplifies into the sampling algorithm of Kim et al. [24]. In retrospect, two main obstacles prevent a straightforward 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 [24]). The second obstacle involves determining how to benefit from a topological order (attribute ordering is irrelevant in [24]); replacing the order with a non-topological one may ruin either the correctness or the efficiency of ADC-sample.

## 4 Subgraph Sampling with Arbitrary Patterns

This section will concentrate on the subgraph sampling problem. As before, let $G = (V, E)$ be the given data graph, which is a simple directed graph where each vertex has an out-degree at most $\lambda$; let $P = (V_P, E_P)$ be the given pattern graph, which is a simple weakly-connected directed graph of a constant size (i.e., $|V_P| = O(1)$). Denote by $occ(G, P)$ the set of occurrences of $P$ in $G$. Our goal is to preprocess $G$ into a data structure that can repeatedly sample from $occ(G, P)$ with low cost.

### 4.1 A Polymatroid Bound on the Number of Occurrences

This subsection will formulate a "polymatroid function" based on $G = (V, E)$ and $P = (V_P, E_P)$ and relate the function to the cardinality of $occ(G, P)$. Our discussion will also clarify why finding the occurrences in $occ(G, P)$ — i.e., the goal of subgraph *listing* — can be achieved using a join.

We will first design a rule collection DC over $V_P$. Recall from Section 2 that a rule collection over $V_P$ comprises triples of the form $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}})$ where $\mathcal{X} \subset \mathcal{Y} \subseteq \mathcal{V}_P$. Setting $m = |E|$, we obtain DC using the procedure below:

Build-DC $(m, \lambda, P)$
0. DC $= \emptyset$
1. **for** each edge $(X, Y) \in E_P$ **do**
2. add $(\emptyset, \{X, Y\}, m)$ to DC
3. add $(\{X\}, \{X, Y\}, \lambda)$ to DC
4. **return** DC

Equipped with $\mathsf{DC}$, we now introduce a function:

$$polymat(m, \lambda, P) \quad = \quad polymat(\mathsf{DC}) \tag{21}$$

where $polymat(\mathsf{DC})$ is as defined in (4).

The above formulation is closely related to a folklore reduction from subgraph listing to joins. To elaborate, let us build a *companion join* $\mathcal{Q}$ from $G$ and $P$ in two steps.

1. For every edge $(X, Y) \in E_P$, add to $\mathcal{Q}$ an empty relation $R_{\{X,Y\}}$ with schema $\{X, Y\}$.
2. For every edge $(X, Y) \in E_P$, insert into the relation $R_{\{X,Y\}}$ a tuple $\boldsymbol{u}$ with $\boldsymbol{u}(X) = x$ and $\boldsymbol{u}(Y) = y$ for each edge $(x, y) \in E$ (i.e., $(x, y)$ is an edge in the data graph $G$).

Importantly, the rule collection $\mathsf{DC}$ acquired from Build-DC serves as a set of degree constraints consistent with $\mathcal{Q}$, i.e., $\mathcal{Q} \models \mathsf{DC}$. Specifically:

- Each triple of the form $(\emptyset, \{X, Y\}, m) \in \mathsf{DC}$ becomes a cardinality constraint stating that $|R_{\{X,Y\}}| \leq m$. It holds because $|R_{\{X,Y\}}|$ is precisely $m$ by our construction.
- Each triple of the form $(\{X\}, \{X, Y\}, \lambda) \in \mathsf{DC}$ becomes a degree constraint stating that $deg_{\{X,Y\}|\{X\}}(R_{\{X,Y\}}) \leq \lambda$. It holds because every vertex $x \in V$ has an out-degree at most $\lambda$ in $G$.

The constraint dependence graph $G_{\mathsf{DC}}$ of $\mathcal{Q}$ is precisely $P$.

The relationships between $join(\mathcal{Q})$ and $occ(G, P)$ satisfy two properties:

- **P1:** Let $\boldsymbol{u}$ be a tuple in $join(\mathcal{Q})$ such that $\boldsymbol{u}$ maps each $X \in V_P$ to a distinct vertex $\boldsymbol{u}(X) \in V$. Then, the subgraph induced by the edge set $\{(\boldsymbol{u}(X), \boldsymbol{u}(Y)) \mid (X, Y) \in E_P\}$ must be an occurrence in $occ(G, P)$ — we say that the occurrence is *produced* by $\boldsymbol{u}$.
- **P2:** Every occurrence in $occ(G, P)$ is produced by the same number $c$ of tuples in $join(\mathcal{Q})$, where $c \geq 1$ is a constant equal to the number of automorphisms of $P$.

If we define

$$\text{OUT} \quad = \quad |occ(G, P)| \tag{22}$$

$$\text{OUT}_{\mathcal{Q}} \quad = \quad |join(\mathcal{Q})| \tag{23}$$

the above discussion implies

$$c \cdot \text{OUT} \leq \text{OUT}_{\mathcal{Q}} \leq polymat(\mathsf{DC}) = polymat(m, \lambda, P). \tag{24}$$

where the second inequality is due to Lemma 2.1. This tells us $\text{OUT} \leq polymat(m, \lambda, P)$. On the other hand, Appendix C proves:

**Lemma 4.1.** *Fix any weakly-connected pattern graph $P = (V_P, E_P)$. For any sufficiently large integers $m$ and $\lambda$ satisfying $\lambda \leq m$, there is a data graph $G = (V, E)$ with $|E| = m$ and maximum vertex out-degree at most $\lambda$ such that $G$ has $\Omega(polymat(m, \lambda, P))$ occurrences of $P$.*

Let us define

$$\mathcal{G}(m, \lambda) \quad = \quad \text{the set of all simple directed graphs } G' \text{ satisfying (i) } G' \text{ has at most } m \text{ edges and (ii) the largest vertex out-degree in } G' \text{ is at most } \lambda. \tag{25}$$

$$\mathcal{F}(m, \lambda, P) \quad = \quad \max_{G' \in \mathcal{G}(m,\lambda)} |occ(G', P)| \tag{26}$$

The above discussion indicates that $polymat(m, \lambda, P)$ is an asymptotically tight characterization of $\mathcal{F}(m, \lambda, P)$, namely, $polymat(m, \lambda, P) = \Theta(\mathcal{F}(m, \lambda, P))$.

### 4.2 Subgraph Sampling and The De-cycling Theorem

Properties **P1** and **P2** in Section 4.1 suggest a reduction from subgraph sampling to join sampling. First, sample a tuple $\boldsymbol{u} \in \mathit{join}(\mathcal{Q})$ uniformly at random. Then, check whether $\boldsymbol{u}(X) = \boldsymbol{u}(Y)$ for any two distinct attributes $X, Y \in V_P$. If so, declare failure; otherwise, declare success and return $\{(\boldsymbol{u}(X), \boldsymbol{u}(Y)) \mid (X, Y) \in E_P\}$ as an occurrence of $P$ in $G$. The success probability equals $c \cdot \mathrm{OUT}/\mathrm{OUT}_{\mathcal{Q}}$, where $c$ the number of automorphisms of $P$, and OUT and $\mathrm{OUT}_{\mathcal{Q}}$ are defined in (22) and (23), respectively. In a success event, every occurrence in $\mathit{occ}(G, P)$ has the same probability to be returned.

As mentioned before, the constraint dependence graph $G_{\mathsf{DC}}$ of $\mathcal{Q}$ is precisely $P$. Therefore, if $P$ is acyclic, so is $G_{\mathsf{DC}}$, in which case our algorithm in Theorem 3.1 can be readily applied to perform subgraph sampling. To analyze the performance, consider first $\mathrm{OUT} \geq 1$. In that scenario, we expect to draw $O(\mathrm{OUT}_{\mathcal{Q}}/\mathrm{OUT})$ samples from $\mathit{join}(\mathcal{Q})$ until having a success event. As Theorem 3.1 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 3.4. That is, besides running the above algorithm for the case $\mathrm{OUT} \geq 1$, we run a concurrent thread that executes Ngo's algorithm in [30], which computes the whole $\mathit{join}(\mathcal{Q})$ and, hence also $\mathit{occ}(G, P)$, 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 constant time. By terminating the whole algorithm as soon as one of the two threads finishes, we ensure $O(\mathit{polymat}(\mathsf{DC})/\max\{1, \mathrm{OUT}\})$ expected time for subgraph sampling.

The main challenge, however, arises when $P$ is cyclic. In this case, $G_{\mathsf{DC}} = P$ is cyclic, meaning that $\mathsf{DC}$ becomes a cyclic set of degree constraints, rendering neither Theorem 3.1 nor Ngo's algorithm in [30] applicable.

The key to overcoming this challenge is the following De-cycling Theorem:

**Theorem 4.2** (De-cycling Theorem)**.** *Fix any pattern graph $P = (V_P, E_P)$ of a constant size. Denote by $d_P^{max}$ the maximum out-degree of a vertex in $P$. Consider any integer $m \geq |E_P|$ and any integer $\lambda \in [d_P^{max}, m]$. Let $\mathsf{DC}$ be the set of degree constraints returned by $\mathsf{Build\text{-}DC}(m, \lambda, P)$. When $\mathsf{DC}$ is cyclic (which happens if and only if $P$ is cyclic), we can always find a proper subset $\mathsf{DC}' \subset \mathsf{DC}$ satisfying (i) $\mathsf{DC}'$ is acyclic, and (ii) $\mathit{polymat}(\mathsf{DC}') = \Theta(\mathit{polymat}(\mathsf{DC}))$.*

The proof is non-trivial and will be presented in the next subsection.

Theorem 4.2 enables us to perform subgraph sampling for a cyclic pattern $P$ directly using Theorem 3.1 and the join algorithm of Ngo [30]. Consider once again the companion join $\mathcal{Q}$ constructed from $G$ and $P$, and let $\mathsf{DC}$ be the output of $\mathsf{Build\text{-}DC}(m, \lambda, P)$. First, obtain a proper subset $\mathsf{DC}'$ of $\mathsf{DC}$ from Theorem 4.2. Because $\mathcal{Q} \models \mathsf{DC}$, we know from $\mathsf{DC}' \subset \mathsf{DC}$ that $\mathcal{Q}$ must be consistent with $\mathsf{DC}'$ as well, i.e., $\mathcal{Q} \models \mathsf{DC}'$. Theorem 3.1 can now be used to extract a sample from $\mathit{join}(\mathcal{Q})$ in $O(\mathit{polymat}(\mathsf{DC}')/\max\{1, \mathrm{OUT}_{\mathcal{Q}}\})$ time. Just as importantly, Theorem 4.2 also permits us to apply Ngo's algorithm in [30] to compute $\mathit{join}(\mathcal{Q})$ in $O(\mathit{polymat}(\mathsf{DC}'))$ time. Therefore, we can now utilize the two-thread approach to sample from $\mathit{occ}(G, P)$ in

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

time.

The astute reader would have noticed that Theorem 4.2 requires the assumptions of $m \geq |E_P|$ and $\lambda \in [d_P^{max}, m]$. To see why these assumptions are harmless, note that when $\lambda$ is outside the range $[d_P^{max}, m]$, the input graph $G$ cannot have any occurrence of $P$ at all. Similarly, if $m < |E_P|$, there can be no occurrence of $P$ in $G$ either. We thus have arrived at:

**Theorem 4.3.** *Let $G = (V, E)$ be a simple directed graph, where each vertex has an out-degree at most $\lambda$. 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 $O(polymat(|E|, \lambda, P)/\max\{1, \text{OUT}\})$ expected time, where* OUT *is the number of occurrences of $P$ in $G$, and $polymat(|E|, \lambda, P)$ is the polymatroid bound in* (21).

**Remarks.** For subgraph listing, Jayaraman et al. [18] described another method to enable the application of Ngo's algorithm [30] to a cyclic $P$. Given the companion join $\mathcal{Q}$, they employ the "degree uniformization" technique [20] to generate $t = O(\text{polylog}\,|E|)$ 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 $\mathsf{DC}_i$ of degree constraints (which may not be a subset of $\mathsf{DC}$) 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, Theorem 4.2 facilitates a *direct* application of Ngo's algorithm to $\mathcal{Q}$, implying that degree uniformization is unnecessary in solving subgraph listing. We believe that this simplification is noteworthy and merits its own dedicated exposition, considering the fundamental and critical nature of subgraph listing. In the absence of Theorem 4.2, integrating our join-sampling algorithm in Theorem 3.1 with the methodology of [18] for the purpose of subgraph sampling would require substantially more effort.

### 4.3 Proof of the De-cycling Theorem (Theorem 4.2)

Let us re-phrase the statement of Theorem 4.2 as follows. Let $P = (V_P, E_P)$ be a *cyclic* pattern graph, $m$ be an integer at least 1, and $\lambda$ an integer at most $m$. Define $\mathsf{DC}$ 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 $G_{\mathsf{DC}}$ is exactly $P$ (and, hence, is cyclic). The objective is to prove the existence of an acyclic $\mathsf{DC}' \subset \mathsf{DC}$ such that $polymat(\mathsf{DC}') = \Theta(polymat(\mathsf{DC}))$.

#### 4.3.1 Case $\lambda > \sqrt{m}$

Let us introduce two variables: $x_{X,Y}$ and $z_{X,Y}$ for every edge $(X, Y)$ in $G_{\mathsf{DC}} = (V_P, E_P)$. For $\lambda > \sqrt{m}$, Jayaraman et al. [18] defined the following LP, which we name $\text{LP}^{(+)}$:

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

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

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

The following result is due to Jayaraman et al. [18]; we provide a proof in Appendix D for self-containment purposes.

**Lemma 4.4.** *The optimal value of LP$^{(+)}$ is at most $O(1) + \log \mathcal{F}(m, \lambda, P)$.*

We have shown in Section 4.1 that $polymat(m, \lambda, P)$ is an asymptotically tight characterization of $\mathcal{F}(m, \lambda, P)$. Hence, by Lemma 4.4, the optimal value of LP$^{(+)}$ is at most

$$O(1) + \log \mathcal{F}(m, \lambda, P) = O(1) + \log \Theta(polymat(m, \lambda, P)) = O(1) + \log polymat(\mathsf{DC}). \tag{27}$$

Next, we establish a crucial lemma.

**Lemma 4.5.** *Any optimal solution to* LP$^{(+)}$ *has the property that the edges in* $\{(X, Y) \in E_P \mid z_{X,Y} > 0\}$ *induce an acyclic subgraph of* $G_{\mathsf{DC}}$.

*Proof.* Consider an arbitrary optimal solution to 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 the graph induced by the edge set 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

$$\begin{aligned}
x'_{A_2,A_3} &= x^*_{A_2,A_3} + z^*_{A_1,A_2} && (28)\\
x'_{A_1,A_2} &= x^*_{A_1,A_2} && (29)\\
z'_{A_2,A_3} &= z^*_{A_2,A_3} - z^*_{A_1,A_2} && (30)\\
z'_{A_1,A_2} &= 0 && (31)
\end{aligned}$$

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 LP$^{(+)}$. However:

$$\begin{aligned}
&\Big(\sum_{(X,Y)\in E_P} x'_{X,Y} \log m + z'_{X,Y} \log \lambda\Big) - \Big(\sum_{(X,Y)\in E_P} x^*_{X,Y} \log m + z^*_{X,Y} \log \lambda\Big)\\
=\ & z^*_{A_1,A_2} \log m - 2 \cdot z^*_{A_1,A_2} \log \lambda\\
<\ & 0 && (32)
\end{aligned}$$

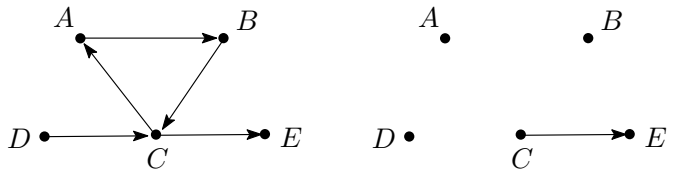
where the last step used the fact $\lambda^2 > m$. This contradicts the optimality of $\{x^*_{X,Y}, z^*_{X,Y} \mid (X, Y) \in E_P\}$. □

We now build a set $\mathsf{DC}'$ of degree constraints as follows.

- First, take an arbitrary optimal solution $\{x^*_{X,Y}, z^*_{X,Y} \mid (X, Y) \in E_P\}$ to LP$^{(+)}$.
- Then, add to $\mathsf{DC}'$ a constraint $(X, \{X, Y\}, \lambda)$ for every $(X, Y) \in E_P$ satisfying $z^*_{X,Y} > 0$.
- Finally, for every edge $(X, Y) \in E_P$, add to $\mathsf{DC}'$ a constraint $(\emptyset, \{X, Y\}, m)$.

Lemma 4.5 assures us that the $\mathsf{DC}'$ thus constructed must be acyclic. Denote by $G_{\mathsf{DC}'} = (V'_P, E'_P)$ the degree constraint graph of $\mathsf{DC}'$. Note that $V_P = V'_P$ and $E'_P \subset E_P$.

*Example 4.1.* We will use the graph $G_{\mathsf{DC}}$ in Figure 4a to illustrate the concepts and methods in this subsection. Here, $V_P = \{A, B, C, D, E\}$, and $E_P = \{(A, B), (B, C), (C, A), (D, C), (C, E)\}$. The LP$^{(+)}$ for this graph is:




(a) $G_{\mathsf{DC}} = P$ (b) $G_{\mathsf{DC}'}$

Figure 4: An example for illustrating the $\lambda > \sqrt{m}$ case

$$\begin{aligned}
\textbf{LP}^{(+)} \quad & \text{minimize } \textstyle\sum_{(X,Y)\in E_P}(x_{X,Y}\log m + z_{X,Y}\log\lambda) \text{ subject to} \\
& x_{A,B} + x_{C,A} + z_{C,A} \geq 1 \\
& x_{A,B} + x_{B,C} + z_{A,B} \geq 1 \\
& x_{B,C} + x_{C,A} + x_{D,C} + x_{C,E} + z_{B,C} + z_{D,C} \geq 1 \\
& x_{D,C} \geq 1 \\
& x_{C,E} + z_{C,E} \geq 1 \\
& x_{X,Y} \geq 0,\ z_{X,Y} \geq 0 \qquad\qquad \forall (X,Y) \in E_P
\end{aligned}$$

An optimal solution to the above LP is

- $x_{A,B} = x_{D,C} = z_{C,E} = 1$,
- the rest variables in $\{x_{X,Y}, z_{X,Y} \mid (X,Y) \in E_P\}$ are set to 0.

The optimal value is $2\log m + \log\lambda$. We can prove that this solution is optimal by the strong duality theorem. Consider the dual LP of $\text{LP}^{(+)}$ for $P$:

$$\begin{aligned}
\textbf{dual LP}^{(+)} \quad & \text{maximize } \textstyle\sum_{X\in V_P}\nu_X \quad \text{subject to} \\
& \nu_X + \nu_Y \leq \log m \qquad \forall (X,Y) \in E_P \\
& \nu_X \leq \log\lambda \qquad \forall X \in V_P \setminus \{D\} \\
& \nu_X \geq 0 \qquad \forall X \in V_P
\end{aligned}$$

We can construct a solution that achieves an objective value of $2\log m + \log\lambda$ by setting $\nu_A = \nu_B = 0.5\log m$, $\nu_C = 0$, $\nu_D = \log m$, and $\nu_E = \log\lambda$. Hence, the solution we constructed is indeed optimal. By our construction, we have $\mathsf{DC}' = \{(\emptyset, \{A,B\}, m), (\emptyset, \{D,C\}, m), (\{C\}, \{C,E\}, \lambda)\}$, and $G_{\mathsf{DC}'}$ is shown in Figure 4b. □

**Lemma 4.6.** *The* $\mathsf{DC}'$ *constructed in the above manner satisfies* $\mathit{polymat}(\mathsf{DC}') = \Theta(\mathit{polymat}(\mathsf{DC}))$.

*Proof.* We will first show that $\mathit{polymat}(\mathsf{DC}') \geq \mathit{polymat}(\mathsf{DC})$. As defined in (4), the calculation of $\mathit{polymat}(\mathsf{DC}')$ involves taking the maximum $h(V_P')$ over all set functions $h \in \Gamma_{V_P'} \cap \mathcal{H}_{\mathsf{DC}'}$. Similarly, the calculation of $\mathit{polymat}(\mathsf{DC})$ involves taking the maximum $h(V_P)$ over all set functions $h \in \Gamma_{V_P} \cap \mathcal{H}_{\mathsf{DC}}$. As $V_P = V_P'$ and $\mathsf{DC}' \subset \mathsf{DC}$, we can assert that $\mathcal{H}_{\mathsf{DC}'}$ must be a superset of $\mathcal{H}_{\mathsf{DC}}$ (because $\mathsf{DC}'$ has fewer constraints than $\mathsf{DC}$). Thus, $\mathit{polymat}(\mathsf{DC}') \geq \mathit{polymat}(\mathsf{DC})$. The rest of the proof will show $\mathit{polymat}(\mathsf{DC}') = O(\mathit{polymat}(\mathsf{DC}))$, which will establish the lemma.

As $\mathsf{DC}'$ is acyclic, the discussion of Section 2 tells us that $\log(\mathit{polymat}(\mathsf{DC}'))$ is the optimal value of the dual modular LP defined by $\mathsf{DC}'$. Next, we will construct a feasible solution to that dual modular LP under which the dual modular LP's objective function equals the optimal value of

$\text{LP}^{(+)}$. Thus, the optimal value of the dual modular LP is at most the optimal value of $\text{LP}^{(+)}$, which is at most $O(1) + \log \mathit{polymat}(\mathsf{DC})$ as shown in (27). As a result, $\mathit{polymat}(\mathsf{DC}') = O(\mathit{polymat}(\mathsf{DC}))$.

Let $\{x^*_{X,Y},\ z^*_{X,Y} \mid (X,Y) \in E_P\}$ denote the optimal solution to $\text{LP}^{(+)}$ from which $\mathsf{DC}'$ was obtained. Recall that 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 assign values to these variables as follows:

- $\delta_{\{X,Y\}|\emptyset} = x^*_{X,Y}$ for each $(X,Y) \in E_P$;
- $\delta_{\{X,Y\}|\{X\}} = z^*_{X,Y}$ for each $(X,Y) \in E'_P$.

First, we prove that the above assignment is a feasible solution to the dual modular LP defined by $\mathsf{DC}'$. By our construction we have $\delta_{\mathcal{Y}|\mathcal{X}} \geq 0$ for each $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}'$. In addition, for every vertex $A \in V_P$, it holds that

$$\begin{aligned}
\sum_{\substack{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}' \\ \text{such that } A\in\mathcal{Y}\setminus\mathcal{X}}} \delta_{\mathcal{Y}|\mathcal{X}} &= \sum_{(X,A)\in E_P} x^*_{X,A} + \sum_{(A,Y)\in E_P} x^*_{A,Y} + \sum_{(X,A)\in E'_P} z^*_{X,A} \\
(\text{since } z^*_{X,A} = 0 \text{ for every } (X,A) \in E_P \setminus E'_P) &= \sum_{(X,A)\in E_P} (x^*_{X,A} + z^*_{X,A}) + \sum_{(A,Y)\in E_P} x^*_{A,Y} \\
(\text{by the first constraint in } \text{LP}^{(+)}) &\geq 1.
\end{aligned}$$

Hence, our assignment is a feasible solution to the dual modular $\text{LP}^{(+)}$ defined by $\mathsf{DC}'$. The objective value of the dual modular LP under the solution constructed is:

$$\begin{aligned}
\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC}'} \delta_{\mathcal{Y}|\mathcal{X}} \cdot \log N_{\mathcal{Y}|\mathcal{X}} &= \sum_{(X,Y)\in E_P} x^*_{X,Y} \log m + \sum_{(X,Y)\in E'_P} z^*_{X,Y} \log \lambda \\
(\text{since } z^*_{X,A} = 0 \text{ for every } (X,A) \in E_P \setminus E'_P) &= \sum_{(X,Y)\in E_P} x^*_{X,Y} \log m + z^*_{X,Y} \log \lambda.
\end{aligned}$$

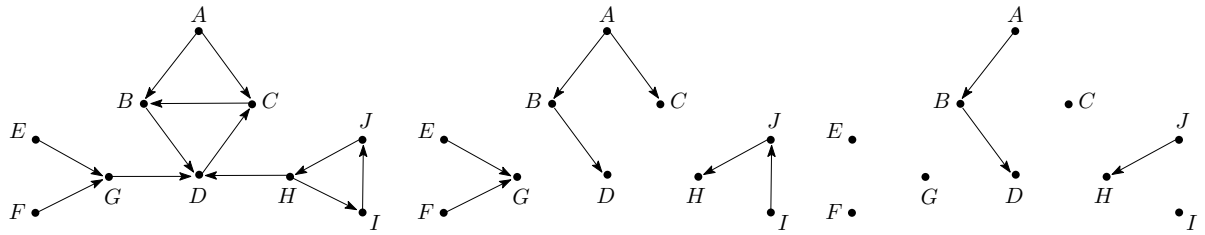
Therefore, the objective value of the dual modular LP is the same as the optimal value of $\text{LP}^{(+)}$, which completes the proof. □

### 4.3.2 Case $\boldsymbol{\lambda \leq \sqrt{m}}$

Let us start by defining some concepts. Recall (from Section 1.2) that the *fractional edge cover number* of a directed graph $G_{\text{dir}}$ — denoted as $\rho(G_{\text{dir}})$ — is defined as the fractional edge cover number of the corresponding undirected graph obtained from $G_{\text{dir}}$ by ignoring the edge directions.

Now consider $G_{\text{dir}}$ to be a directed bipartite graph $G_{bp} = (V_{bp}, E_{bp})$. By Lemma 8.2 of [32] (see also Lemma 3.1 of [5] and Theorem 30.10 of [35]), it is always possible to decompose $G_{bp}$ into vertex-disjoint subgraphs $\star_1, \star_2, ...,$ and $\star_\alpha$ (for some integer $\alpha > 0$) such that

- $\star_i$ is a directed star for each $i \in [\alpha]$ (a directed star is a graph consisting of $t \geq 2$ vertices, among which one vertex, called 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);
- $\sum_{i=1}^{\alpha} \rho(\star_i) = \rho(G_{bp})$.




(a) $G_{\mathsf{DC}} = P$ (b) $G^*$ (c) $G_{\mathsf{DC}'}$

Figure 5: An example for illustrating the $\lambda \leq \sqrt{m}$ case

We will refer to $\{\star_1, \star_2, ..., \star_\alpha\}$ as a *fractional edge-cover decomposition* of $G_{bp}$. It is worth mentioning that the fractional edge-cover decomposition in Lemma 8.2 of [32] also comprises "odd-length cycles". However, when $G_{bp}$ is a bipartite graph, the decomposition includes only directed stars.

Next, we review a formula from [18] that, as we will prove later, is $O(\mathcal{F}(m, \lambda, P))$ where $\mathcal{F}(m, \lambda, P)$ is defined in (26). Find all the strongly connected components (SCCs) of $G_{\mathsf{DC}} = (V_P, E_P)$ (recall that $G_{\mathsf{DC}}$ is the same as the pattern graph $P$). Call an SCC a *source SCC* if it has no in-coming edge from another SCC. Furthermore, a source SCC is (i) *trivial* if it has a single vertex, or (ii) *non-trivial* otherwise.

*Example 4.2.* We will use the graph $G_{\mathsf{DC}}$ in Figure 5a to illustrate the concepts and methods in this subsection. Here, $V_P = \{A, B, ..., J\}$, and $E_P$ includes the 12 edges shown. There are 6 SCCs: $\{A\}$, $\{B, C, D\}$, $\{E\}$, $\{F\}$, $\{G\}$, and $\{H, I, J\}$. Among them, $\{A\}$, $\{E\}$, $\{F\}$, and $\{H, I, J\}$ are source SCCs. Furthermore, $\{A\}$, $\{E\}$, and $\{F\}$ are trivial source SCCs, while $\{H, I, J\}$ is a non-trivial source SCC. □

Define:

$$\mathcal{S} = \text{the set of vertices in } G_{\mathsf{DC}} \text{ each forming a trivial source SCC by itself} \quad (33)$$

$$\mathcal{T} = \text{the set of vertices in } G_{\mathsf{DC}} \text{ receiving an in-coming edge from at least one vertex in } \mathcal{S} \quad (34)$$

The sets $\mathcal{S}$ and $\mathcal{T}$ must be disjoint because each vertex $\mathcal{T}$ has an in-coming edge and thus cannot belong to a source SCC. Take a fractional edge-cover decomposition $\Sigma^*$ of the directed bipartite graph induced by $\mathcal{S}$ and $\mathcal{T}$; as mentioned, $\Sigma^*$ is a set of directed stars.

Define:

$$\mathcal{I} = \text{the set of vertices in the non-trivial source SCCs of } G_{\mathsf{DC}} \quad (35)$$

$$\mathcal{J} = \mathcal{V}_P \setminus (\mathcal{S} \cup \mathcal{T} \cup \mathcal{I}) \quad (36)$$

It is worth pointing out that $\mathcal{S}, \mathcal{T}, \mathcal{I}$, and $\mathcal{J}$ are mutually disjoint. We now introduce three quantities:

$$n_1 = |\mathcal{I}| \quad (37)$$

$$n_2 = |\mathcal{J}| = |V_P| - n_1 - |\mathcal{S}| - |\mathcal{T}|. \quad (38)$$

$$n_3 = \text{number of non-trivial source SCCs} \quad (39)$$

$$n_4 = \text{the number of edges in the directed stars of } \Sigma^* \quad (40)$$

*Example 4.3.* Consider again the graph $G_{\mathsf{DC}}$ in Figure 5a. It has one non-trivial source SCC $\{H, I, J\}$, giving $n_3 = 1$. Also, $\mathcal{S} = \{A, E, F\}$ and $\mathcal{T} = \{G, B, C\}$. The directed bipartite graph

induced by $\mathcal{S}$ and $\mathcal{T}$ has 4 edges: $(A, B)$, $(A, C)$, $(E, G)$ and $(F, G)$. A fractional edge-cover cover $\Sigma^*$ of this bipartite graph has two directed stars: the first has center $A$ and petals $B$ and $C$, while the second has center $G$ and petals $E$ and $F$. Thus, $n_4 = 4$. Set $\mathcal{I}$ includes $H$, $I$, and $J$, while $\mathcal{J} = \{D\}$. Hence, $n_1 = 3$, and $n_2 = 1 = 10 - 3 - 3 - 3$. □

The lemma below was claimed in [18] but we are unable to verify their proof. In Appendix E, we provide our own proof of the statement.

**Lemma 4.7.** *Fix a pattern graph $P = (V_P, E_P)$. For any integer $m \geq |E_P|$ and integer $\lambda \in [d_P^{max}, \sqrt{m}]$, we have $\mathcal{F}(m, \lambda, P) = \Omega(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2+n_4-2n_3-|\mathcal{S}|})$.*

Because (as proved in Section 4.1) $polymat(m, \lambda, P)$ is $\Theta(\mathcal{F}(m, \lambda, P))$, it follows that

$$polymat(m, \lambda, P) \quad = \quad \Omega\Big(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2+n_4-2n_3-|\mathcal{S}|}\Big). \tag{41}$$

Next, we construct the acyclic degree constraint set $\mathsf{DC}'$ that fulfills the requirement in our de-cycling theorem (no such construction was given in [18]). Let $G^* = (V^*, E^*)$ be an arbitrary weakly-connected acyclic subgraph of $G_{\mathsf{DC}}$ satisfying the conditions below.

- $V^* = V_P$.
- $E^*$ contains all the edges in the directed stars of $\Sigma^*$.
- In every non-trivial source SCC, each vertex — with a single exception which we identify the SCC's *root* $X_{root}$ — has exactly one in-coming edge in $E^*$. No in-coming edge of $X_{root}$ is in $E^*$ but $E^*$ includes at least one out-going edge of $X_{root}$. We designate one arbitrary out-going edge of $X_{root}$ in $E^*$ as the SCC's *main edge*.
- Every vertex in $\mathcal{J}$ has exactly one in-coming edge included in $E^*$.

It is rudimentary to build such a subgraph $G^*$, e.g., using depth first traversal. The existence of $G^*$ is guaranteed by the fact that $G_{\mathsf{DC}}$ is weakly connected.

*Example 4.4.* Let us build $G^* = (V^*, E^*)$ for the graph $G_{\mathsf{DC}}$ in Figure 5a. The first bullet sets $V^* = V_P = \{A, B, ..., J\}$. The second bullet adds edges $(A, B), (A, C), (E, G)$, and $(F, G)$ to $E^*$. The third bullet concerns the (only) non-trivial source SCC $\{H, I, J\}$. We add edges $(J, H)$ and $(I, J)$ to $E^*$. Here, $I$ is the root of the SCC, and $(I, J)$ is the main edge of the SCC. The last bullet concerns the only vertex $D$ in $\mathcal{J}$; adding edge $(B, D)$ to $E^*$ fulfill the bullet's requirement. This completes the construction of $E^* = \{(A, B), (A, C), (E, G), (F, G), (J, H), (I, J), (B, D)\}$. The final $G^*$ is shown in Figure 5b. □

Equipped with $G_{\mathsf{DC}} = (V_P, E_P)$ and $G^* = (V^*, E^*)$, we now create a set $\mathsf{DC}'$ of degree constraints in four steps:

- For each edge $(X, Y) \in E_P$, add a constraint $(\emptyset, \{X, Y\}, m)$ to $\mathsf{DC}'$.
- Inspect each directed star in $\Sigma^*$ and distinguish two scenarios.
  - Scenario 1: either the star has only one edge or its center is from $\mathcal{T}$. Nothing needs to be done in this case.

  - Scenario 2: The star has more than one edge and its center $X$ is from $\mathcal{S}$. In this case, pick an arbitrary petal $Y$ and designate $(X, Y)$ as the star's *main edge*. Then, for every other petal $Y'$, add a constraint $(\{X\}, \{X, Y'\}, \lambda)$ to $\mathsf{DC}'$.

- Next, consider each non-trivial source SCC. Remember that every vertex $Y$, other than the SCC's root, has an in-coming edge $(X, Y) \in E^*$. For every such $Y$, if $(X, Y)$ is not the SCC's main edge, add a constraint $(\{X\}, \{X, Y\}, \lambda)$ to $\mathsf{DC}'$.

- Finally, recall that every vertex $Y \in \mathcal{J}$ has an in-coming edge $(X, Y)$ in $E^*$; add a constraint $(\{X\}, \{X, Y\}, \lambda)$ to $\mathsf{DC}'$.

By the above construction, the constraint dependency graph $G_{\mathsf{DC}'}$ of $\mathsf{DC}'$ is a subgraph of $G^*$ and, hence, must be acyclic (because $G^*$ is acyclic). The set $\mathsf{DC}'$ we have obtained is thus acyclic.

*Example 4.5.*Continuing on the previous example, we now construct $\mathsf{DC}'$ from the $G_{\mathsf{DC}}$ and $G^*$ in Figures 5a and 5b. The first bullet adds to $\mathsf{DC}'$ the constraints $(\emptyset, AB, m)$, $(\emptyset, AC, m)$, $(\emptyset, CB, m)$, $(\emptyset, BD, m)$, $(\emptyset, DC, m)$, $(\emptyset, EG, m)$, $(\emptyset, FG, m)$, $(\emptyset, GD, m)$, $(\emptyset, HD, m)$, $(\emptyset, JH, m)$, $(\emptyset, HI, m)$, and $(\emptyset, IJ, m)$. The second bullet inspects the two stars in $\Sigma^*$. The first star has center $A$ and pellets $B$ and $C$. Because $A \in \mathcal{S}$ and the star contains two petals, we fall into Scenario 2. Suppose that we designate $(A, C)$ as the star's main edge; accordingly, this necessitates adding the constraint $(A, AB, \lambda)$ to $\mathsf{DC}'$. The second star has center $G$ and pellets $E$ and $F$. As $G \in \mathcal{T}$, we fall into Scenario 1, where no constraints are added to $\mathsf{DC}'$. The third bullet processes the non-trivial source SCC $\{H, I, J\}$. As mention in Example 4.4, this SCC has root $I$; furthermore, $E^*$ has an in-coming edge $(I, H)$ of $H$ and an in-coming edge $(I, J)$ of $J$. For $(J, H)$, we add $(J, JH, \lambda)$ to $\mathsf{DC}'$; for $(I, J)$, however, we add nothing because it is the main edge of the SCC (see Example 4.4). The last bullet processes the sole vertex $D$ in $\mathcal{J}$. After identifying its (only) in-coming edge $(B, D)$ in $E^*$, we add $(B, BD, \lambda)$ to $\mathsf{DC}'$.

The final $\mathsf{DC}'$ is therefore

$$\{(\emptyset, AB, m),\ (\emptyset, AC, m),\ (\emptyset, CB, m),\ (\emptyset, BD, m),\ (\emptyset, DC, m),\ (\emptyset, EG, m),\ (\emptyset, FG, m),\ (\emptyset, GD, m),\\ (\emptyset, HD, m),\ (\emptyset, JH, m),\ (\emptyset, HI, m),\ (\emptyset, IJ, m),\ (A, AB, \lambda),\ (J, JH, \lambda),\ (B, BD, \lambda)\}.$$

The constraint dependency graph $G_{\mathsf{DC}'}$ is shown in Figure 5c. □

The rest of the proof will show $\mathit{polymat}(\mathsf{DC}') = \Theta(\mathit{polymat}(\mathsf{DC}))$. Since $\mathsf{DC}' \subset \mathsf{DC}$, we have $\mathit{polymat}(\mathsf{DC}') \geq \mathit{polymat}(\mathsf{DC})$ (the reason behind this can be found in the proof of Lemma 4.6). It remains to show that $\mathit{polymat}(\mathsf{DC}') = O(\mathit{polymat}(\mathsf{DC}))$. As $\mathsf{DC}'$ is acyclic, the discussion of Section 2 tells us that $\log(\mathit{polymat}(\mathsf{DC}'))$ is the optimal value of the dual modular LP defined by $\mathsf{DC}'$. Next, we will construct a feasible solution to that dual modular LP under which the LP's objective function equals

$$\Big((n_3 + |\mathcal{S}|) \cdot \log m\Big) + (n_1 + n_2 + n_4 - 2n_3 - |\mathcal{S}|) \cdot \log \lambda. \tag{42}$$

As the optimal value of the dual modular LP cannot exceed (42), it follows that $\mathit{polymat}(\mathsf{DC}')$ is $O(m^{c_1+|\mathcal{S}|} \cdot \lambda^{n_1+n_2+n_4-2n_3-|\mathcal{S}|})$, which is $O(\mathit{polymat}(m, \lambda, P)) = O(\mathit{polymat}(\mathsf{DC}))$ due to (41).

Recall that 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 according to the rules below.

- For every constraint $(\mathcal{X}, \mathcal{Y}, \lambda) \in \mathsf{DC}'$, set $\delta_{\mathcal{Y}|\mathcal{X}} = 1$.

- Consider each directed star in $\Sigma^*$ and again distinguish two cases.
  - Scenario 1: The star contains only one edge or its center is in $\mathcal{T}$. In this case, set $\delta_{\{X,Y\}|\emptyset} = 1$ for every edge $(X, Y)$ in the star.
  - Scenario 2: The star contains more than one edge and has a center $X \in \mathcal{S}$. Recall that the set $\mathsf{DC}'$ has a constraint $(\emptyset, \{X, Y\}, m)$ for the star's main edge $(X, Y)$. Set $\delta_{\{X,Y\}|\emptyset} = 1$.
- Consider each non-trivial source SCC. Recall that $\mathsf{DC}'$ contains a constraint $(\emptyset, \{X, Y\}, m)$ for the main edge $(X, Y)$ of the SCC. Set $\delta_{\{X,Y\}|\emptyset} = 1$.

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

*Example 4.6.* For our running example, the variable $\delta_{\mathcal{Y}|\mathcal{X}}$ of each $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}'$ is decided as follows. The first bullet sets $\delta_{AB|A}$, $\delta_{JH|J}$, and $\delta_{BD|B}$ to 1. The second bullet inspects the two directed stars in $\Sigma^*$. The first directed star has center $A \in \mathcal{S}_1$ (recall from Example 4.3 that $\mathcal{S}_1 = \{A\}$) and petals $\{B, C\}$. As the star's main edge is $(A, C)$ (designated in Example 4.5), we set $\delta_{AC|\emptyset} = 1$. The second directed star of $\Sigma^*$ has center $G \in \mathcal{T}_1$ (recall from Example 4.3 that $\mathcal{T}_1 = \{G\}$) and petals $\{E, F\}$. Thus, we set $\delta_{GE|\emptyset} = \delta_{GF|\emptyset} = 1$. The third bullet examines the non-trivial source SCC $\{H, J, I\}$, whose main edge is $(I, J)$ (as decided in Example 4.4). We thus set $\delta_{IJ|\emptyset} = 1$. The remaining variables $\delta_{AB|\emptyset}$, $\delta_{BC|\emptyset}$, $\delta_{BD|\emptyset}$, $\delta_{CD|\emptyset}$, $\delta_{GD|\emptyset}$, $\delta_{HD|\emptyset}$, $\delta_{JH|\emptyset}$, and $\delta_{HI|\emptyset}$ are all set to 0. □

It is straightforward to verify that all the constraints of the dual modular LP are satisfied. To confirm that the objective function indeed evaluates to (42), observe:

- There are $n_3 + |\mathcal{S}|$ constraints of the form $(\emptyset, \{X, Y\}, m)$ with $\delta_{\{X,Y\}|\emptyset} = 1$, where $\mathcal{S}$ and $n_3$ are defined in (33) and (39), respectively. Specifically, $n_3$ of them come from the roots of the non-trivial source SCCs, and $|\mathcal{S}|$ of them come from the directed stars in $\Sigma^*$ (combining Scenarios 1 and 2).
- There are $n_1 + n_2 + n_4 - 2n_3 - |\mathcal{S}|$ of the form $(\{X\}, \{X, Y\}, \lambda)$ with $\delta_{\{X,Y\}|\{X\}} = 1$, where $n_1$, $n_2$, and $n_4$ are defined in (37), (38), and (40), respectively. Specifically, (i) $n_1 - 2n_3$ of them come from the non-main edges of the non-trivial source SCCs, (ii) $n_4 - |\mathcal{S}|$ of them come from the non-main edges in the stars that have a center vertex in $S$ and have at least two edges, and (iii) $n_2$ of them come from the vertices in $\mathcal{J}$.

*Example 4.7.* We will demonstrate that the variable values chosen in Example 4.6 fulfill all the constraints of the dual modular LP. Clearly, all the variable values are non-negative. Thus, it remains to verify the following for each vertex $Z \in V_P$:

$$\sum_{\substack{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}})\in\mathsf{DC} \\ \text{such that } Z\in\mathcal{Y}\setminus\mathcal{X}}} \delta_{\mathcal{Y}|\mathcal{X}} \geq 1.$$

For $Z = A$ (resp., $C, E, F, G, I$, and $J$), the above holds because $\delta_{AC|\emptyset}$ (resp., $\delta_{AC|\emptyset}$, $\delta_{GE|\emptyset}$, $\delta_{GF|\emptyset}$, $\delta_{GF|\emptyset}$, $\delta_{IJ|\emptyset}$, and $\delta_{IJ|\emptyset}$) equals 1. For $Z = B$ (resp., $H$ and $D$), the above holds because $\delta_{AB|A}$ (resp., $\delta_{JH|J}$ and $\delta_{BD|B}$) equals 1.

There are $n_3 + |\mathcal{S}| = 1 + 3 = 4$ constraints of the form $(\emptyset, \{X, Y\}, m)$ that satisfy $\delta_{\{X,Y\}|\emptyset} = 1$: they are $(\emptyset, AB, m)$, $(\emptyset, AC, m)$, $(\emptyset, GD, m)$, and $(\emptyset, FG, m)$. Moreover, $n_1 + n_2 + n_4 - 2n_3 - |\mathcal{S}| =$

$4 + 3 + 1 - 2 - 3 = 3$ constraints of the form $(\{X\}, \{X, Y\}, \lambda)$ satisfy $\delta_{\{X,Y\}|\{X\}} = 1$: they are $(J, JH, \lambda)$, $(A, AB, \lambda)$, and $(B, BD, \lambda)$. Among them, (i) $n_1 - 2n_3 = 1$ constraint — $(J, JH, \lambda)$ — comes from the (only) non-main edge of the non-trivial source SCC $\{H, I, J\}$, (ii) $n_4 - |\mathcal{S}| = 1$ constraint — $(A, AB, \lambda)$ — comes from the petal $B$ of the star whose center is $A$, and (iii) $n_2 = 1$ constraint — $(B, BD, \lambda)$ — comes from the vertex $D \in \mathcal{J}$.

With the above variable values, the dual modular LP's objective function evaluates to $4 \log m + 3 \log \lambda$, i.e., the value in (42). □

We now conclude the whole proof of Theorem 4.2.

# 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 $\mathsf{DC}$ 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 $\epsilon$ with high probability (i.e., at least $1 - 1/\text{IN}^c$ for an arbitrarily large constant $c$) in time $\tilde{O}(\frac{1}{\epsilon^2} \frac{\mathit{polymat}(\mathsf{DC})}{\max\{1,\text{OUT}\}})$ after a preprocessing of $O(\text{IN})$ expected time.
- Employing a technique in [13], we can, with high probability, report all the tuples in $\mathit{join}(\mathcal{Q})$ with a delay of $\tilde{O}(\frac{\mathit{polymat}(\mathsf{DC})}{\max\{1,\text{OUT}\}})$. 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 $\mathit{join}(\mathcal{Q})$ 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 $\mathit{polymat}(\mathsf{DC})$ over $\mathit{AGM}(\mathcal{Q})$. In addition, our findings in the last two bullet points also complement Ngo's algorithm as described in [30] in a satisfying manner.

# Appendix

# A Linear Programming

The material of this section can be found in most textbooks (e.g., [27]) on linear programming (LP) and is included for self-containment reasons.

Suppose that $\mathbf{A}$ is an $n \times m$ matrix, $\boldsymbol{c}$ is an $n \times 1$ matrix (a.k.a., an $n$-dimensional vector), and $\boldsymbol{b}$ an $m \times 1$ matrix (a.k.a., an $m$-dimensional vector). Consider an LP of the form:

$$\text{find an } n \times 1 \text{ matrix } \boldsymbol{x} \text{ to maximize } \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \text{ subject to } \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b} \text{ and } \boldsymbol{x} \geq 0 \tag{43}$$

where $\boldsymbol{x} \geq 0$ means that every component of $\boldsymbol{x}$ must be at least 0. The duality of the above LP has the form:

$$\text{find an } m \times 1 \text{ matrix } \boldsymbol{y} \text{ to minimize } \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \text{ subject to } \mathbf{A}^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c} \text{ and } \boldsymbol{y} \geq 0. \tag{44}$$

We will refer to (43) as the *primal form* and to (44) as the *dual form*.

The *strong duality theorem* states that the primal form returns a finite optimal value if and only if the dual form returns the same optimal value.

The optimal $\boldsymbol{x}$ maximizing $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ — we will refer to $\boldsymbol{x}$ as an *optimal solution* to the LP — in the primal LP has a geometric property. Let us regard $\boldsymbol{x}$ as an $n$-dimensional point in $\mathbb{R}^n$. Then, the constraints $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$ and $\boldsymbol{x} \geq 0$ define a *feasible region* for the point $\boldsymbol{x}$. The feasible region is a *polyhedron*, formally defined as the intersection of a finite number of halfspaces in $\mathbb{R}^n$. The polyhedron, in general, can be unbounded. However, if the LP has a finite optimal value, then an optimal solution to the LP can be found at a vertex of the polyhedron.

Similar geometric interpretations also apply to the dual LP, except that $\boldsymbol{y}$ should be regarded as an $m$-dimensional point in $\mathbb{R}^m$. The feasible region is a polyhedron defined by $\mathbf{A}^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c}$ and $\boldsymbol{y} \geq 0$. If the LP has a finite optimal value, then an optimal solution $\boldsymbol{y}$ can be found at a vertex of the polyhedron.

We will also need the *complementary slackness theorem*. Let us write out the detailed components of $\boldsymbol{b}$ and $\boldsymbol{c}$ as $\boldsymbol{b} = (b_1, b_2, ..., b_m)$ and $\boldsymbol{c} = (c_1, c_2, ..., c_n)$. Denote by $\boldsymbol{u}_i$ the $i$-th row of $\mathbf{A}$ and by $\boldsymbol{v}_j$ the $j$-th column of $\mathbf{A}$; note that $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are $n$- and $m$-dimensional vectors, respectively. Assume that $\boldsymbol{x}^* = (x_1^*, x_2^*, ..., x_n^*)$ is an optimal solution to the primal LP (43). The complementary slackness theorem states that the dual LP (44) has an optimal solution $\boldsymbol{y}^* = (y_1^*, y_2^*, ..., y_m^*)$ satisfying the following conditions:

- If $x_j^* > 0$, then $\boldsymbol{v}_j^{\mathrm{T}} \cdot \boldsymbol{y} = c_j$ for $j \in [n]$.
- If $\boldsymbol{v}_j^{\mathrm{T}} \cdot \boldsymbol{y} > c_j$, then $x_j^* = 0$ for $j \in [n]$.
- If $y_i^* > 0$, then $\boldsymbol{u}_i^{\mathrm{T}} \cdot \boldsymbol{x}^* = b_i$ for $i \in [m]$.
- If $\boldsymbol{u}_i^{\mathrm{T}} \cdot \boldsymbol{x}^* < b_i$, then $y_i^* = 0$ for $i \in [m]$.

# B Implementing ADC-Sample with Indexes

Recall that $A_1, A_2, ..., A_k$ form a topological order of the attributes in $G_{\mathsf{DC}}$. As defined in (7), $\mathcal{V}_0 = \emptyset$ and $V_i = \{A_1, ..., A_i\}$ for $i \geq 1$.

We preprocess each constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$ as follows. Let $R \in Q$ be its main guard, i.e., $R = R_{F(\mathcal{X},\mathcal{Y})}$. For each $i \in [k]$ and each tuple $\boldsymbol{w} \in \Pi_{schema(R)\cap\mathcal{V}_i}(R)$, define

$$R_{\mathcal{Y}}(i, \boldsymbol{w}) \quad = \quad \{\boldsymbol{u}[\mathcal{Y}] \mid \boldsymbol{u} \in R, \boldsymbol{u}[schema(R) \cap \mathcal{V}_i] = \boldsymbol{w}\}$$

which we refer to as a *fragment*.

During preprocessing, we compute and store $R_{\mathcal{Y}}(i, \boldsymbol{w})$ for every $i \in [k]$ and $\boldsymbol{w} \in \Pi_{schema(R)\cap\mathcal{V}_i}(R)$. Next, we will explain how to do so for an arbitrary $i \in [k]$. First, group all the tuples of $R$ by the attributes in $schema(R) \cap \mathcal{V}_i$, which can be done in $O(\mathrm{IN})$ expected time by hashing. Then, perform the following steps for each group in turn. Let $\boldsymbol{w}$ be the group's projection on $schema(R) \cap \mathcal{V}_i$. We compute the group tuples' projections onto $\mathcal{Y}$ and eliminate duplicate projections, the outcome of which is precisely $R_{\mathcal{Y}}(i, \boldsymbol{w})$ and is stored using an array. With hashing, this requires expected time linear to the group's size. Therefore, the total cost of generating the fragments $R_{\mathcal{Y}}(i, \boldsymbol{w})$ of all $\boldsymbol{w} \in \Pi_{schema(R)\cap\mathcal{V}_i}(R)$ is $O(\mathrm{IN})$ expected. We also build a hash table such that given any $i \in [k]$

and tuple $\boldsymbol{w} \in \Pi_{schema(R)\cap\mathcal{V}_i}(R)$, we can retrieve the starting address and size of $R_{\mathcal{Y}}(i, \boldsymbol{w})$ in $O(1)$ time. The cost of building this hash table is $O(\text{IN})$ expected.

After the above preprocessing, given any $i \in [k]$, constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$, tuple $\boldsymbol{w}$ over $\mathcal{V}_{i-1}$, and value $v \in \mathbf{dom}$, we can compute $\mathit{reldeg}_{\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, v)$ defined in (8) in constant time. For convenience, let $R = R_{F(\mathcal{X},\mathcal{Y})}$. To compute $|\Pi_{\mathcal{Y}}(R \ltimes \boldsymbol{w})|$ (the denominator of (8)), first obtain $\boldsymbol{w}_1 = \boldsymbol{w}[\mathit{schema}(R) \cap \mathcal{V}_{i-1}]$. Then, $\Pi_{\mathcal{Y}}(R \ltimes \boldsymbol{w})$ is just the fragment $R_{\mathcal{Y}}(i-1, \boldsymbol{w}_1)$, which has been pre-stored. The size of this fragment can be retrieved using $\boldsymbol{w}_1$ in $O(1)$ time. Similarly, to compute $|\sigma_{A_i=v}(\Pi_{\mathcal{Y}}(R \ltimes \boldsymbol{w}))|$ (the numerator of (8)), we can first obtain $\boldsymbol{w}_2$, which is a tuple over $\mathit{schema}(R) \cap \mathcal{V}_i$ that shares the values of $\boldsymbol{w}_1$ on all the attributes in $\mathit{schema}(R) \cap \mathcal{V}_{i-1}$ and additionally uses value $v$ on attribute $A_i$. Then, $\sigma_{A_i=v}(\Pi_{\mathcal{Y}}(R \ltimes \boldsymbol{w}))$ is just the fragment $R_{\mathcal{Y}}(i, \boldsymbol{w}_2)$, which has been pre-stored. The size of this fragment can be fetched using $\boldsymbol{w}_2$ in $O(1)$ time.

As a corollary, given any $i \in [k]$, tuple $\boldsymbol{w}$ over $\mathcal{V}_{i-1}$, and value $v \in \mathbf{dom}$, we can compute $\mathit{reldeg}^*(\boldsymbol{w}, v)$ and $\mathit{constraint}^*(\boldsymbol{w}, v)$ — defined in (9) and (10), respectively — in constant time.

It remains to explain how to implement Line 4 of ADC-sample (Figure 3). Here, we want to randomly sample a tuple from $\Pi_{\mathcal{Y}^\circ}(R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)} \ltimes \boldsymbol{w}_{i-1})$. Again, for convenience, let $R = R_{F(\mathcal{X}^\circ,\mathcal{Y}^\circ)}$. Obtain $\boldsymbol{w}' = \boldsymbol{w}_{i-1}[\mathit{schema}(R) \cap \mathcal{V}_{i-1}]$. Then, $\Pi_{\mathcal{Y}}(R \ltimes \boldsymbol{w}_{i-1})$ is just the fragment $R_{\mathcal{Y}}(i-1, \boldsymbol{w}')$, which has been stored in an array. The starting address and size of the array can be acquired using $\boldsymbol{w}'$ in $O(1)$ time, after which a sample can be drawn from the fragment in constant time.

# C Proof of Lemma 4.1

Recall that $P = (V_P, E_P)$ is a directed graph. Let $P' = (V_{P'}, E_{P'})$ be the undirected counterpart of $P$, namely, $V_{P'} = V_P$ and $E_{P'}$ contains an (undirected) edge $\{X, Y\}$ if and only if $(X, Y) \in E_P$ or $(Y, X) \in E_P$. Set $m' = m/|E_P|$ and $\lambda' = \lambda/|E_P|$. Let $\mathsf{DC}'$ be the rule collection output by Build-DC$(m', \lambda', P)$.

The collection $\mathsf{DC}'$ is a type of "simple" degree constraints as classified by Suciu [37], who proved the existence of a joint $\mathcal{Q}$ satisfying:

- the schema graph of $\mathcal{Q}$ is $P'$;
- each relation of $\mathcal{Q}$ has at most $m'$ tuples;
- $|\mathit{join}(\mathcal{Q})| = \Omega(\mathit{polymat}(\mathsf{DC}'))$.

For each attribute $X \in V_P$, we use $\mathbf{actdom}(X)$ to denote the *active domain* of $X$ in $\mathcal{Q}$, which includes all the $X$-values appearing in at least one relation of $\mathcal{Q}$, or formally:

$$\mathbf{actdom}(X) = \{x \in \mathbf{dom} \mid \exists R \in \mathcal{Q}, \boldsymbol{u} \in R : X \in \mathit{schema}(R), \boldsymbol{u}(X) = x\}.$$

We assume that the attributes have disjoint active domains, namely, $\mathbf{actdom}(X) \cap \mathbf{actdom}(Y) = \emptyset$ for any different $X, Y \in V$. This loses no generality because we can always prefix each value with the name of its attribute.

Now, construct a graph $G = (V, E)$ as follows:

- $V = \bigcup_{X \in V} \mathbf{actdom}(X)$.
- Consider each relation $R \in \mathcal{Q}$, and suppose that it has schema $\{X, Y\}$. For each tuple $\boldsymbol{u} \in R$, we add to $E$ (i) an edge from vertex $\boldsymbol{u}(X)$ to vertex $\boldsymbol{u}(Y)$ if $(X, Y) \in E_P$, and (ii) an edge from vertex $\boldsymbol{u}(Y)$ to vertex $\boldsymbol{u}(X)$ if $(Y, X) \in E_P$.

It is clear that $G$ can have at most $m' \cdot |\mathcal{Q}| = m' \cdot |E_P| = m$ edges. Furthermore, each vertex in $G$ can have a degree at most $\lambda' \cdot |E_P| = \lambda$. To explain why, let us consider an arbitrary vertex $x \in V$, which let us assume is a value from $\mathbf{actdom}(X)$. If $x$ has an out-neighbor $y \in V$ — which let us assume comes from $\mathbf{actdom}(Y)$ — then (i) some relation $R \in \mathcal{Q}$ must contain a tuple $\boldsymbol{u}$ with $\boldsymbol{u}(X) = x$ and $\boldsymbol{u}(Y) = y$, and (ii) $(X, Y)$ is an edge in $E_P$. As $\mathsf{DC}'$ contains a degree constraint $(\{X\}, \{X, Y\}, \lambda')$, there can be at most $\lambda'$ different choices for $y$. It thus follows that the degree of $x$ in $G$ cannot exceed $\lambda' \cdot |\mathcal{Q}| = \lambda' \cdot |E_P| = \lambda$.

Next, we argue that $G$ has at least $|\mathit{join}(\mathcal{Q})| = \Omega(\mathit{polymat}(\mathsf{DC}'))$ occurrences of $P$. Consider an arbitrary tuple $\boldsymbol{u} \in \mathit{join}(\mathcal{Q})$. It is easy to verify that, for any edge $(X, Y) \in E_P$, there exists an edge $(\boldsymbol{u}(X), \boldsymbol{u}(Y))$ in $G$. Hence, $G$ contains an occurrence with the edge set $\{(\boldsymbol{u}(X), \boldsymbol{u}(Y)) \mid (X, Y) \in E_P\}$. As no two tuples in $\mathit{join}(\mathcal{Q})$ correspond to the same occurrence (their occurrences must differ in at least one vertex), the number of occurrences must be at least $|\mathit{join}(\mathcal{Q})|$.

It remains to show that $\mathit{polymat}(\mathsf{DC}') = \Omega(\mathit{polymat}(\mathsf{DC}))$. This is a corollary of Lemma C.1 that we will establish next.

## C.1 A Property of Polymatroid Bounds under Degree Constraints

This subsection serves as a proof of the following lemma.

**Lemma C.1.** *Fix a positive constant $\alpha \geq 1$. Consider any join $\mathcal{Q}$ with the schema graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set $\mathsf{DC}$ of degree constraints. Construct another set $\mathsf{DC}'$ of degree constraints as follows:*

$$\mathsf{DC}' = \{(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}/\alpha) \mid (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}\}. \tag{45}$$

*Then, we must have*

$$\mathit{polymat}(\mathsf{DC}') \geq \rho \cdot \mathit{polymat}(\mathsf{DC}) \tag{46}$$

*where $\rho$ is a positive value that depends only on $G_{\mathsf{DC}}$ (the constraint dependency graph), and the constant $\alpha$.*

Given a constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$, we will refer to $N_{\mathcal{Y}|\mathcal{X}}$ as the constraint's *threshold*. The lemma indicates that the value $\rho$ has nothing to do with the threshold fields of the constraints in $\mathsf{DC}$.

The starting point of our proof is the observation that the polymatroid bound on a degree-constraint set is the optimal value of an LP. To compute $\mathit{polymat}(\mathsf{DC})$, for example, we need to find a set function $h \in \Gamma_{\mathcal{V}}$ (see Section 2 for the definition of $\Gamma_{\mathcal{V}}$) maximizing $h(\mathcal{V})$ while satisfying $h(\mathcal{Y}) - h(\mathcal{X}) \leq \log N_{\mathcal{Y}|\mathcal{X}}$ for every $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$. We can view $h$ as a vector $\boldsymbol{h}$ in a $2^{|\mathcal{V}|}$-dimensional space where each "dimension" corresponds to a distinct subset $\mathcal{X}$ of $\mathcal{V}$. Accordingly, the "coordinate" of $\boldsymbol{h}$ on the dimension $\mathcal{X}$ equals the value of $h(\mathcal{X})$. Thus, $\log(\mathit{polymat}(\mathsf{DC}))$ is the value returned by the following LP:

$$\text{find an } n \times 1 \text{ matrix } \boldsymbol{h} \text{ to maximize } \boldsymbol{c}^{\mathrm{T}}\boldsymbol{h} \text{ subject to } \mathbf{A}\boldsymbol{h} \leq \boldsymbol{b} \text{ and } \boldsymbol{h} \geq 0 \tag{47}$$

where $n = 2^{|\mathcal{V}|}$, $\boldsymbol{c}$ is a "one-hot" vector with coordinate 1 on the dimension $\mathcal{V}$ but coordinate 0 on all other dimensions, and $\mathbf{A}\boldsymbol{h} \leq \boldsymbol{b}$ captures the following constraints:

$$\text{(zero-grounded)} \quad h(\emptyset) \leq 0 \tag{48}$$

$$\text{(monotone)} \quad h(\mathcal{X}) - h(\mathcal{Y}) \leq 0 \quad \forall \mathcal{X}, \mathcal{Y} \text{ satisfying } \mathcal{X} \subset \mathcal{Y} \subseteq \mathcal{V} \tag{49}$$

$$\text{(submodular)} \quad h(\mathcal{X} \cup \mathcal{Y}) + h(\mathcal{X} \cap \mathcal{Y}) - h(\mathcal{X}) - h(\mathcal{Y}) \leq 0 \quad \forall \mathcal{X}, \mathcal{Y} \subseteq \mathcal{V} \tag{50}$$

$$\text{(degree constraint)} \quad h(\mathcal{Y}) - h(\mathcal{X}) \leq \log N_{\mathcal{Y}|\mathcal{X}} \quad \forall (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC} \tag{51}$$

We do not need to calculate precisely the number $m$ of rows in $\mathbf{A}$ except to note that

- $m$ is a finite value dependent on $\mathcal{V}$ and $|\mathsf{DC}|$, and
- every row corresponds to a distinct constraint listed above.

The following facts are immediate:

- **F1:** the matrix $\mathbf{A}$ has nothing to do with the threshold fields of the constraints in $\mathsf{DC}$, but is determined by $G_{\mathsf{DC}}$ (the constraint dependency graph).
- **F2:** The vector $\boldsymbol{b}$ takes value $\log N_{\mathcal{Y}|\mathcal{X}}$ at each row corresponding to a constraint of the form (51), and value 0 at all other rows.
- **F3:** The LP's optimal value $\log(\mathit{polymat}(\mathsf{DC}))$ is finite because $\mathit{polymat}(\mathsf{DC}) \leq \mathit{AGM}(\mathcal{Q})$.

The LP in (47) is in the primal form (the reader may wish to revisit Appendix A before proceeding). Its dual counterpart is:

$$\text{LP}_{\mathsf{DC}}\text{: find an } m \times 1 \text{ matrix } \boldsymbol{y} \text{ to minimize } \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \text{ subject to } \mathbf{A}^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c} \text{ and } \boldsymbol{y} \geq 0. \tag{52}$$

The following observations apply to $\text{LP}_{\mathsf{DC}}$:

- The vector $\boldsymbol{y}$ is in $\mathbb{R}^m$, where each dimension corresponds to a distinct constraint in (48)-(51). The feasible region of $\text{LP}_{\mathsf{DC}}$ is the polyhedron below:

$$\Psi = \{\boldsymbol{y} \mid \mathbf{A}^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c} \text{ and } \boldsymbol{y} \geq 0\}. \tag{53}$$

- The optimal value of $\text{LP}_{\mathsf{DC}}$ is $\log(\mathit{polymat}(\mathsf{DC}))$, i.e., same as the primal LP, due to fact **F3** and the strong duality theorem.
- Let $\boldsymbol{y}_{\mathsf{DC}}$ be an optimal solution to $\text{LP}_{\mathsf{DC}}$. The point $\boldsymbol{y}_{\mathsf{DC}}$ is a vertex of the feasible region. If we denote by $\boldsymbol{y}_{\mathsf{DC}}[\mathcal{Y}|\mathcal{X}]$ the coordinate of $\boldsymbol{y}_{\mathsf{DC}}$ on the dimension corresponding to the constraint $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$, facts **F2** and **F3** tell us:

$$\log(\mathit{polymat}(\mathsf{DC})) = \sum_{(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} \boldsymbol{y}_{\mathsf{DC}}[\mathcal{Y}|\mathcal{X}] \cdot \log N_{\mathcal{Y}|\mathcal{X}}. \tag{54}$$

Let us now turn our attention to $\mathit{polymat}(\mathsf{DC}')$. By the same reasoning, we know that $\log(\mathit{polymat}(\mathsf{DC}'))$ is the optimal value of the following LP:

$$\text{find an } n \times 1 \text{ matrix } \boldsymbol{h} \text{ to maximize } \boldsymbol{c}^{\mathrm{T}}\boldsymbol{h} \text{ subject to } \mathbf{A}\boldsymbol{h} \leq \boldsymbol{b'} \text{ and } \boldsymbol{h} \geq 0 \tag{55}$$

where

- the vector $\boldsymbol{c}$ is the same as that in (47);
- $\mathbf{A}\boldsymbol{h} \leq \boldsymbol{b'}$ captures the constraints (48)-(51) except that the value $\log N_{\mathcal{Y}|\mathcal{X}}$ in (51) is now replaced with $\log(N_{\mathcal{Y}|\mathcal{X}}/\alpha)$

- the matrix $\mathbf{A}$ is the same as that in (47).

The dual form of (55) is:

$$\text{LP}_{\mathsf{DC}'}\text{: find an } m \times 1 \text{ matrix } \boldsymbol{y} \text{ to minimize } \boldsymbol{b'}^{\mathrm{T}}\boldsymbol{y} \text{ subject to } \mathbf{A}^{\mathrm{T}}\boldsymbol{y} \geq \boldsymbol{c} \text{ and } \boldsymbol{y} \geq 0. \tag{56}$$

Note that the feasible region of $\text{LP}_{\mathsf{DC}'}$ is the same polyhedron $\Psi$ in $\mathbb{R}^m$ given by (53).

The optimal value of $\text{LP}_{\mathsf{DC}'}$ is $\log(\mathit{polymat}(\mathsf{DC}'))$. If $\boldsymbol{y}_{\mathsf{DC}'}$ be an optimal solution to $\text{LP}_{\mathsf{DC}'}$, then — just like (54) — we have:

$$\log(\mathit{polymat}(\mathsf{DC}')) \quad = \quad \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}/\alpha)\in\mathsf{DC}'} \boldsymbol{y}_{\mathsf{DC}'}[\mathcal{Y}|\mathcal{X}] \cdot \log(N_{\mathcal{Y}|\mathcal{X}}/\alpha)$$

which yields

$$\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}/\alpha)\in\mathsf{DC}'} \boldsymbol{y}_{\mathsf{DC}'}[\mathcal{Y}|\mathcal{X}] \cdot \log N_{\mathcal{Y}|\mathcal{X}} = \log(\mathit{polymat}(\mathsf{DC}')) + \log\alpha \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}/\alpha)\in\mathsf{DC}'} \boldsymbol{y}_{\mathsf{DC}'}[\mathcal{Y}|\mathcal{X}]. \tag{57}$$

Because the optimal value of $\text{LP}_{\mathsf{DC}'}$ is finite, the point $\boldsymbol{y}_{\mathsf{DC}'}$ must be a vertex of the polyhedron $\Psi$ in (53). Let us introduce the quantity below obtained by examining all vertices of $\Psi$:

$$\beta \quad = \quad \max_{\text{vertex } \boldsymbol{y} \text{ of } \Psi} \sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}/\alpha)\in\mathsf{DC}'} \boldsymbol{y}[\mathcal{Y}|\mathcal{X}].$$

From (53) and fact **F1**, we know that $\beta$ depends only on the constraint dependency graph $G_{\mathsf{DC}}$ of $\mathsf{DC}$. It thus follows from (57) that

$$\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}/\alpha)\in\mathsf{DC}'} \boldsymbol{y}_{\mathsf{DC}'}[\mathcal{Y}|\mathcal{X}] \cdot \log N_{\mathcal{Y}|\mathcal{X}} \leq \log(\mathit{polymat}(\mathsf{DC}')) + \beta\log\alpha. \tag{58}$$

Finally, we relate $\text{LP}_{\mathsf{DC}'}$ to $\text{LP}_{\mathsf{DC}}$. Starting from the fact that $\boldsymbol{y}_{\mathsf{DC}'}$ may not be an optimal solution to $\text{LP}_{\mathsf{DC}}$, we derive:

$$\begin{aligned}
\boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}_{\mathsf{DC}'} &\geq \log(\mathit{polymat}(\mathsf{DC})) \Rightarrow \\
\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}/\alpha)\in\mathsf{DC}'} \boldsymbol{y}_{\mathsf{DC}'}[\mathcal{Y}|\mathcal{X}] \cdot \log N_{\mathcal{Y}|\mathcal{X}} &\geq \log(\mathit{polymat}(\mathsf{DC})) \Rightarrow \\
\text{(using (58))} \quad \log(\mathit{polymat}(\mathsf{DC}')) + \beta\log\alpha &\geq \log(\mathit{polymat}(\mathsf{DC})) \Rightarrow \\
\mathit{polymat}(\mathsf{DC}') &\geq \mathit{polymat}(\mathsf{DC})/(2^{\beta}\cdot\alpha).
\end{aligned}$$

Thus, setting $\rho = 1/(2^{\beta}\cdot\alpha)$ completes the proof of Lemma C.1.

# D Proof of Lemma 4.4

As in the statement of Theorem 4.2, we use $d_P^{max}$ to represent the maximum out-degree of a vertex in $P$. Furthermore, the values of $m$ and $\lambda$ satisfy $m \geq |E_P|$ and $\lambda \geq d_P^{max}$.

We consider the dual of $\text{LP}^{(+)}$:

| | | |
|---|---|---|
| **dual LP**$^{(+)}$ | maximize $\sum_{X\in\mathcal{V}} \nu_X$ subject to | |
| (I) | $\nu_X + \nu_Y \leq \log m$ | for each $(X,Y) \in E_P$ |
| (II) | $\nu_Y \leq \log\lambda$ | for each $(X,Y) \in E_P$ |
| (III) | $\nu_X \geq 0$ | for each $X \in V_P$ |

Let $\{\nu_X^* | X \in V_P\}$ be an optimal solution to the dual $\text{LP}^{(+)}$. By the strong duality theorem, the optimal value of the $\text{LP}^{(+)}$ is $\sum_{X \in V_P} \nu_X^*$.

Next, we will show that there exists a data graph $G$ satisfying the following properties:

- $G$ has at most $m$ edges.
- The out-degree of each vertex in $G$ is at most $\lambda$;
- $G$ contains $\Omega(\exp_2(\sum_{X \in V_P} \nu_X^*))$ occurrences of $P$.

The number of occurrence of $P$ in $G$ cannot exceed $\mathcal{F}(m, \lambda, P)$. Hence, the existence of $G$ indicates $\exp_2(\sum_{X \in V_P} \nu_X^*) = O(\mathcal{F}(m, \lambda, P))$, meaning that $\sum_{X \in V_P} \nu_X^*$ — the optimal value of the $\text{LP}^{(+)}$ — is at most $O(1) + \log \mathcal{F}(m, \lambda, P)$, as claimed in the lemma.

We will construct the desired graph $G = (V, E)$ as follows.

- For each vertex $X \in V_P$, create a vertex set $V_X$ with $\max\{\lfloor \exp_2(\nu_X^*)/|E_P| \rfloor, 1\}$ vertices. After that, set $V$ (the vertex set of $G$) to $\bigcup_{X \in V_P} V_X$.
- For each edge $(X, Y) \in E_P$, create an edge set $E_{X,Y} = \{(u, v) | u \in V_X, v \in V_Y\}$. After that, set $E$ (the edge set of $G$) to $\bigcup_{(X,Y) \in E_P} E_{X,Y}$.

For each $(X, Y) \in E_P$, the number of edges between $V_X$ and $V_Y$ is

$$|V_X| \cdot |V_Y| = \max\{\lfloor \exp_2(\nu_X^*)/|E_P| \rfloor, 1\} \cdot \max\{\lfloor \exp_2(\nu_Y^*)/|E_P| \rfloor, 1\},$$

which is at most the maximum between the four values below:

- 1
- $\lfloor \exp_2(\nu_X^*)/|E_P| \rfloor$
- $\lfloor \exp_2(\nu_Y^*)/|E_P| \rfloor$, and
- $\lfloor \exp_2(\nu_X^*)/|E_P| \rfloor \cdot \lfloor \exp_2(\nu_Y^*)/|E_P| \rfloor$.

**Proposition D.1.** *All the four values above are bounded by $m/|E_P|$.*

*Proof.* First, $1 \leq m/|E_P|$ due to $m \geq |E_P|$. Second, $\lfloor \exp_2(\nu_X^*)/|E_P| \rfloor \leq \exp_2(\nu_X^* + \nu_Y^*)/|E_P| \leq m/|E_P|$, where the first inequality is because $\nu_Y^* \geq 0$, and the second inequality is by the constraint set (I) in the dual $\text{LP}^{(+)}$. Third, $\lfloor \exp_2(\nu_Y^*)/|E_P| \rfloor \leq \exp_2(\nu_X^* + \nu_Y^*)/|E_P| \leq m/|E_P|$, where the first inequality is because $\nu_X^* \geq 0$. Finally, $\lfloor \exp_2(\nu_X^*)/|E_P| \rfloor \cdot \lfloor \exp_2(\nu_Y^*)/|E_P| \rfloor \leq \exp_2(\nu_X^* + \nu_Y^*)/|E_P| \leq m/|E_P|$. □

Therefore, the total number of edges in $G$ is at most $|E_P| \cdot m/|E_P| = m$.

The out-degree of each vertex $X$ in $G$ is $\sum_{(X,Y) \in E_P} |V_Y| = \sum_{(X,Y) \in E_P} \max\{\lfloor \exp_2(\nu_Y^*)/|E_P| \rfloor, 1\}$, which is at most the maximum between $d_P^{max}$ and

$$\begin{aligned} & d_P^{max} \cdot \max_{(X,Y) \in E_P} \exp_2(\nu_Y^*)/|E_P| \\ \text{(by constraint set (II) of } LP^{(+)}) \quad \leq \quad & d_P^{max} \cdot \exp_2(\log \lambda)/|E_P|, \\ = \quad & \lambda \cdot d_P^{max}/|E_P| \end{aligned}$$

$$\leq \quad \lambda$$

Hence, the maximum degree of the vertex in $G$ is at most $\lambda$.

An occurrence of $P$ in $G$ can be formed by mapping each vertex $X \in V_P$ to an arbitrary vertex in $V_X$. Hence, the number of occurrences of $P$ in $G$ is at least

$$\begin{aligned}
\prod_{X \in V_P} \max\{1, \lfloor \exp_2(\nu^*_X)/|E_P| \rfloor\} &\geq \prod_{X \in V_P} \exp_2(\nu^*_X)/(2|E_P|) \\
&= \exp_2\Big(\sum_{X \in V_P} \nu^*_X\Big)/(2|E_P|)^{|V_P|} \\
&= \Omega\Big(\exp_2\Big(\sum_{X \in V_P} \nu^*_X\Big)\Big)
\end{aligned}$$

which completes the proof.

# E Proof of Lemma 4.7

We will show that there exists a data graph $G = (V, E)$ satisfying the following conditions:

- $G$ has at most $m$ edges;
- The out-degree of each vertex in $G$ is at most $\lambda$.
- $G$ has $\Omega(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2-2n_3+n_4-|\mathcal{S}|})$ occurrences of $P$.

As $G \in \mathcal{G}(m, \lambda)$ (see the definition in (25)), the number of occurrences of $P$ in $G$ cannot exceed $\mathcal{F}(m, \lambda, P)$ (see the definition in (26)). The existence of $G$ implies $\mathcal{F}(m, \lambda, P) = \Omega(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2-2n_3+n_4-|\mathcal{S}|})$.

**Bipartite Fractional Edge-Cover Decompositions.** Before constructing $G$, we make a connection between the fractional edge-cover decomposition and the vertex-pack LP. Denote the directed bipartite subgraph induced by $\mathcal{S}$ and $\mathcal{T}$ as $\mathcal{G}_{bp} = (\mathcal{S} \cup \mathcal{T}, \mathcal{E}_{bp})$, where $\mathcal{E}_{bp}$ includes all edges in $E_P$ between the vertices in $\mathcal{S}$ and those in $\mathcal{T}$ (every such edge must be directed from $\mathcal{S}$ to $\mathcal{T}$ by the definitions of $\mathcal{S}$ and $\mathcal{T}$). Consider the following vertex-pack LP for $\mathcal{G}_{bp}$:

**verex-pack LP** maximize $\sum_{X \in \mathcal{S} \cup \mathcal{T}} \nu_X$ subject to

$$\begin{aligned}
&\nu_X + \nu_Y \leq 1 && \text{for each } (X, Y) \in \mathcal{E}_{bp} \\
&\nu_X \geq 0 && \text{for each } X \in \mathcal{S} \cup \mathcal{T}
\end{aligned}$$

We will prove the next lemma in Section E.1.

**Lemma E.1.** *Given any fractional edge-cover decomposition $\Sigma^*$ of $\mathcal{G}_{bp}$, we can find an optimal integral solution $\{\nu^*_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ to the above vertex-pack LP satisfying:*

1. *$\nu^*_X = 0$ or $1$ for each $X \in \mathcal{S} \cup \mathcal{T}$.*
2. *For each directed star in $\Sigma^*$ having at least two edges, $\nu^*_X = 0$ for the center vertex $X$ and $\nu^*_Y = 1$ for each petal vertex $Y$.*
3. *For each directed star in $\Sigma^*$ with only one edge $(X, Y)$, $\nu^*_X + \nu^*_Y = 1$.*

**Basic Definitions and Useful Inequalities.** Fix an optimal solution $\{\nu_X^* \mid X \in \mathcal{S} \cup \mathcal{T}\}$ promised by Lemma E.1. Define

$$\begin{aligned}
\mathcal{S}_1 &= \{X \in \mathcal{S} \mid \nu_X^* = 0\} \\
\mathcal{S}_2 &= \mathcal{S} \setminus \mathcal{S}_1 \\
\mathcal{T}_1 &= \{X \in \mathcal{T} \mid \nu_X^* = 0\} \\
\mathcal{T}_2 &= \mathcal{T} \setminus \mathcal{T}_1.
\end{aligned}$$

Conditions 2 and 3 of Lemma E.1 tell us that $\Sigma^*$ has exactly $|\mathcal{S}_2| + |\mathcal{T}_2|$ edges, and thus

$$|\mathcal{T}_2| - |\mathcal{S}_1| = |\mathcal{T}_2| + |\mathcal{S}_2| - (|\mathcal{S}_2| + |\mathcal{S}_1|) = n_4 - |\mathcal{S}|. \tag{59}$$

Recall (from the statement of Theorem 4.2) that $d_P^{max}$ represents the maximum out-degree of a vertex in $P$; furthermore, the values of $m$ and $\lambda$ satisfy $m \geq |E_P|$ and $\lambda \geq d_P^{max}$. In the subsequent discussion, we assume $m \geq 4|E_P|^2$ (otherwise, $\lambda \leq m = O(1)$ such that $m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2+n_4-2n_3-|\mathcal{S}|} = O(1)$, in which case we can construct the desired $G$ as $G = P$).

Define:

$$\begin{aligned}
m' &= \lfloor m/|E_P| \rfloor \\
\lambda' &= \lfloor \lambda / d_P^{max} \rfloor \\
\lambda^* &= \begin{cases} \lambda' & \text{if } m' \geq \lambda'^2 \\ \lfloor \sqrt{m'} \rfloor & \text{otherwise} \end{cases} \\
s &= \max\{\lfloor m'/\lambda'^2 \rfloor, 1\}
\end{aligned}$$

Note that $m', \lambda', \lambda^*$, and $s$ are at least 1.

**Proposition E.2.** *The following inequalities are true:*

$$s \cdot \lambda^{*2} \leq m' \tag{60}$$

$$\lambda^* \cdot d_P^{max} \leq \lambda \tag{61}$$

$$\lambda < m' \tag{62}$$

$$\lfloor \sqrt{m'} \rfloor = \Omega(\lambda) \tag{63}$$

*Furthermore, if $m' < \lambda'^2$, then*

$$m/\lambda^2 < 2|E_P|/(d_P^{max})^2. \tag{64}$$

*Proof.* Inequality (60) holds because $s \cdot \lambda^{*2} \leq \max\{\lfloor m'/\lambda'^2 \rfloor \cdot \lambda'^2, (\lfloor \sqrt{m'} \rfloor)^2\} \leq m'$. Inequality (61) holds because $\lambda^* \cdot d_P^{max} \leq \lambda' \cdot d_P^{max} \leq \lambda$. Inequality (62) holds because

$$\begin{aligned}
m' &> m/|E_P| - 1 \\
(\text{as } \lambda \leq \sqrt{m} \text{ in Lemma 4.7}) &\geq \lambda \cdot \sqrt{m}/|E_P| - 1 \\
(\text{as } m \geq 4|E_P|^2) &\geq \lambda \cdot \sqrt{4|E_P|^2}/|E_P| - 1 \\
&= 2\lambda - 1
\end{aligned}$$

which is at least $\lambda$ because $\lambda \geq 1$.

To prove (63), first note that

$$\sqrt{m'} = \sqrt{\lfloor m/|E_P| \rfloor} \geq \sqrt{\lfloor 4|E_P| \rfloor} \geq 2$$

where the first inequality used $m \geq 4|E_P|^2$. Hence

$$\begin{aligned} \lfloor \sqrt{m'} \rfloor &\geq \sqrt{m'}/2 = \sqrt{\lfloor m/|E_P| \rfloor}/2 \\ (\text{as } m \geq |E_P|) &\geq \sqrt{m/(2|E_P|)}/2 \\ (\text{as } m \geq \lambda^2) &\geq \frac{\lambda/2}{\sqrt{2|E_P|}} = \Omega(\lambda). \end{aligned}$$

Finally, Inequality (64) follows from the derivation below:

$$\begin{aligned} & m' < \lambda'^2 \\ \Rightarrow \quad & \frac{m}{|E_P|} - 1 < \Big(\frac{\lambda}{d_P^{max}}\Big)^2 \\ (\text{as } m \geq 4|E_P|^2) \quad \Rightarrow \quad & \frac{m}{2|E_P|} < \Big(\frac{\lambda}{d_P^{max}}\Big)^2 \end{aligned}$$

which yields $m/\lambda^2 < 2|E_P|/(d_P^{max})^2$. □

**Construction of $G$.** Next, we explain how to create the desired graph $G = (V, E)$. The construction of $V$ starts with three steps.

- For each vertex $X \in \mathcal{I}$ (see the definition of $\mathcal{I}$ in (35)), create $s$ sets of vertices — denoted as $V_X^1, V_X^2, ..., V_X^s$ — each having $\lambda^*$ vertices. Let $V_X$ be the union of these $s$ sets; thus, $|V_X| = s \cdot \lambda^*$.
- For each vertex $X \in \mathcal{S}_1$, create a set $V_X$ of $\lfloor m'/\lambda^* \rfloor$ vertices. We have $|V_X| \geq 1$ because of (62) and $\lambda^* \leq \lambda$. For each vertex $X \in \mathcal{S}_2$, create a set $V_X$ of $m'$ vertices.
- For each vertex $X \in \mathcal{T}_1$, create a singleton set $V_X$ having only one vertex. For each vertex $X \in \mathcal{T}_2 \cup \mathcal{J}$ (see the definition of $\mathcal{J}$ in (36)), create a set $V_X$ of $\lambda^*$ vertices.

Then, $V = \bigcup_{X \in V_P} V_X$.

To create the edge set $E$ of $G$, we process each edge $(X, Y)$ of $P$ as follows:

- If $X \in \mathcal{I}$ and $Y \in \mathcal{I}$, build a set $E_{X,Y}^i$ — for each $i \in [s]$ — containing an edge from every vertex in $V_X^i$ to every vertex in $V_Y^i$. Define $E_{X,Y} = \bigcup_{i=1}^s E_{X,Y}^i$.
- Otherwise, build a set $E_{X,Y}$ containing an edge from every vertex in $V_X$ to every vertex in $V_Y$.

Then, $E = \bigcup_{(X,Y) \in E_P} E_{X,Y}$.

**Number of Edges in $G$.** Note that $E_P$ cannot contain edges of the following types:

(a) an incoming edge to a vertex in $\mathcal{S}$;

(b) an edge from a vertex in $\mathcal{S}$ to a vertex in $\mathcal{J}$;

(c) an edge from a vertex in $\mathcal{S}_2$ to a vertex in $\mathcal{T}_2$.

Edges of types (a) and (b) cannot exist due to the definitions of $\mathcal{S}$ and $\mathcal{J}$. Assume that an edge $(X, Y)$ of type (c) exists in $E_P$. By the definitions of $\mathcal{S}_2$ and $\mathcal{T}_2$, we know that $\nu_X^* \neq 0$ and $\nu_Y^* \neq 0$. Thus, condition 1 of Lemma E.1 tells us $\nu_X^* = \nu_Y^* = 1$. However, $\nu_X^* + \nu_Y^* = 2 > 1$ violates the first constraint of the vertex-pack LP.

It thus follows that every edge of $E_P$ belongs to one of the following 6 categories:

(1) the edge is from a vertex in $\mathcal{I}$ to another vertex in $\mathcal{I}$ (both vertices must be in the same SCC of $P$);

(2) the edge is from a vertex in $\mathcal{S}_1$ to a vertex in $\mathcal{T}_2$;

(3) the edge is from a vertex in $\mathcal{S}_2$ to a vertex in $\mathcal{T}_1$;

(4) the edge is from a vertex in $\mathcal{S}_1$ to a vertex in $\mathcal{T}_1$;

(5) the edge is from a vertex in $\mathcal{I}$ to a vertex in $\mathcal{T} \cup \mathcal{J}$.

(6) the edge is from a vertex in $\mathcal{T} \cup \mathcal{J}$ to another vertex in $\mathcal{T} \cup \mathcal{J}$.

We are ready to count the number of edges in $G$. Recall that $E$ is the union of $E_{X,Y}$ of all $(X,Y) \in E_P$. We will show that, for every $(X,Y) \in E_P$, the size of $E_{X,Y}$ is bounded by $m'$. As a result, the number of edges in $G$ is $\sum_{(X,Y)\in E_P} |E_{X,Y}| \leq |E_P| \cdot m' \leq m$, as desired.

We will analyze $|E_{X,Y}|$ based on the category of $(X,Y)$. For each category-(1) edge $(X,Y)$,

$$|E_{X,Y}| = \sum_{i\in[s]} |V_X^i| \cdot |V_Y^i| \quad = \quad s \cdot \lambda^{*2}$$

which is at most $m'$ by (60). For an edge $(X,Y) \in E_P$ of category (2)-(6), the size of $E_{X,Y}$ is $|V_X| \cdot |V_Y|$. Specifically:

- For category (2), $|E_{X,Y}| = \lfloor m'/\lambda^* \rfloor \cdot \lambda^* \leq m'$.
- For category (3), $|E_{X,Y}| = m' \cdot 1 = m'$.
- For category (4), $|E_{X,Y}| = \lfloor m'/\lambda^* \rfloor \cdot 1 \leq m'$.
- For category (5), $|E_{X,Y}| = (s \cdot \lambda^*) \cdot \lambda^* \leq m'$, where the inequality is due to (60)
- For category (6), $|E_{X,Y}| \leq \lambda^* \cdot \lambda^* \leq m'$, where the last inequality is due to the definition of $\lambda^*$.

**Maximum Out-Degree of $G$.** Next, we will verify that the out-degree of each vertex in $G$ is at most $\lambda$. For each vertex in the set $V_X^i$ where $X \in \mathcal{I}$ and $i \in [s]$, its out-degree is

$$\sum_{\substack{(X,Y)\in E_P \text{ s.t. } X \text{ and } Y \\ \text{are in the same SCC}}} |V_Y^i| + \sum_{\substack{(X,Y)\in E_P \\ \text{s.t. } Y\in\mathcal{T}\cup\mathcal{J}}} |V_Y| \quad \leq \quad \lambda^* \cdot d_P^{max} \leq \lambda$$

where the last inequality used (61). Now, consider a vertex $v \in V_X$ where $X \in V_P \setminus \mathcal{I}$. As each out-neighbor of $X$ is in $\mathcal{T} \cup \mathcal{J}$, the out-degree of $v$ is

$$\sum_{\substack{(X,Y)\in E_P \\ \text{s.t. } Y\in\mathcal{T}\cup\mathcal{J}}} |V_Y| \leq \lambda^* \cdot d_P^{max} \leq \lambda.$$

**Number of Occurrences of $P$ in $G$.** To complete the proof of Lemma 4.7, it remains to show that $G$ has many occurrences of $P$. We can form a distinct occurrence of $P$ in $G$ as follows:

- for each non-trivial source SCC, pick an integer $i \in [s]$ and then map each vertex $X$ in this SCC to an arbitrary vertex in $V_X^i$;
- for each vertex $X \in V_P \setminus \mathcal{I}$, map $X$ to an arbitrary vertex in $V_X$.

Hence, the number of occurrences of $P$ in $G$ is at least

$$s^{n_3} \cdot (\lambda^*)^{n_1} \cdot (\lfloor \frac{m'}{\lambda^*} \rfloor)^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot (\lambda^*)^{|\mathcal{T}_2|} \cdot (\lambda^*)^{n_2}. \tag{65}$$

If $m' \geq \lambda'^2$, (65) is equal to

$$\begin{aligned}
& (\lfloor \frac{m'}{\lambda'^2} \rfloor)^{n_3} \cdot (\lambda')^{n_1} \cdot (\lfloor \frac{m'}{\lambda'} \rfloor)^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot (\lambda')^{|\mathcal{T}_2|} \cdot (\lambda')^{n_2} \\
(\text{as } m'/\lambda'^2 \geq 1 \text{ implies } m'/\lambda' \geq 1) \quad \geq \quad & (\frac{m'}{2\lambda'^2})^{n_3} \cdot (\lambda')^{n_1} \cdot (\frac{m'}{2\lambda'})^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot (\lambda')^{|\mathcal{T}_2|} \cdot (\lambda')^{n_2} \\
= \quad & 2^{-n_3-|\mathcal{S}_1|} \cdot (m')^{n_3+|\mathcal{S}|} \cdot (\lambda')^{n_1-2n_3-|\mathcal{S}_1|+|\mathcal{T}_2|+n_2} \\
(\text{as } m' = \Omega(m), \lambda' = \Omega(\lambda)) \quad = \quad & \Omega(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2-2n_3+|\mathcal{T}_2|-|\mathcal{S}_1|}) \\
(\text{by (59)}) \quad = \quad & \Omega(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2-2n_3+n_4-|\mathcal{S}|}).
\end{aligned}$$

If $m' < \lambda'^2$, (65) is equal to

$$\begin{aligned}
& 1^{n_3} \cdot (\lfloor \sqrt{m'} \rfloor)^{n_1} \cdot (\lfloor \frac{m'}{\lfloor \sqrt{m'} \rfloor} \rfloor)^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot (\lfloor \sqrt{m'} \rfloor)^{|\mathcal{T}_2|} \cdot (\lfloor \sqrt{m'} \rfloor)^{n_2} \\
(\text{by (64)}) \quad \geq \quad & (\frac{m \cdot (d_P^{max})^2}{\lambda^2 \cdot 2|E_P|})^{n_3} \cdot (\lfloor \sqrt{m'} \rfloor)^{n_1} \cdot (\lfloor \frac{m'}{\sqrt{m'}} \rfloor)^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot (\lfloor \sqrt{m'} \rfloor)^{|\mathcal{T}_2|} \cdot (\lfloor \sqrt{m'} \rfloor)^{n_2} \\
(\text{as } \lfloor \sqrt{m'} \rfloor \geq \sqrt{m'}/2) \quad = \quad & \Omega\big((\frac{m}{\lambda^2})^{n_3} \cdot (\sqrt{m'})^{n_1} \cdot (\frac{m'}{\sqrt{m'}})^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot (\sqrt{m'})^{|\mathcal{T}_2|} \cdot (\sqrt{m'})^{n_2}\big) \\
(\text{by (63)}) \quad = \quad & \Omega\big((\frac{m}{\lambda^2})^{n_3} \cdot \lambda^{n_1} \cdot (\frac{m'}{\sqrt{m'}})^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot \lambda^{|\mathcal{T}_2|} \cdot \lambda^{n_2}\big) \\
(\text{by } \sqrt{m'} \leq \lambda' \leq \lambda) \quad = \quad & \Omega\big((\frac{m}{\lambda^2})^{n_3} \cdot \lambda^{n_1} \cdot (\frac{m'}{\lambda})^{|\mathcal{S}_1|} \cdot (m')^{|\mathcal{S}_2|} \cdot \lambda^{|\mathcal{T}_2|} \cdot \lambda^{n_2}\big) \\
(\text{by } m' = \Omega(m)) \quad = \quad & \Omega(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2-2n_3+|\mathcal{T}_2|-|\mathcal{S}_1|}) \\
(\text{by (59)}) \quad = \quad & \Omega(m^{n_3+|\mathcal{S}|} \cdot \lambda^{n_1+n_2-2n_3+n_4-|\mathcal{S}|}).
\end{aligned}$$

We now complete the proof of Lemma 4.7.

## E.1 Proof of Lemma E.1

The dual of the vertex-pack LP is the following "edge-cove LP":

**edge-cover LP** minimize $\sum_{e \in \mathcal{E}_{bp}} w_e$ subject to

$$\begin{aligned}
& \sum_{e \in \mathcal{E}_{bp}, X \in e} w_e \geq 1 && \text{for each } X \in \mathcal{S} \cup \mathcal{T} \\
& w_e \geq 0 && \text{for each } e \in \mathcal{E}_{bp}
\end{aligned}$$

Let $\{w_e^* \mid e \in \mathcal{E}_{bp}\}$ be the fractional edge-cover of $\mathcal{G}_{bp}$ corresponding to $\Sigma^*$, namely, $w_e^* = 1$ for each edge $e$ in $\Sigma^*$, and $w_e^* = 0$ for each edge $e \in \mathcal{E}_{bp} \setminus \Sigma^*$. By the definition of fractional edge-cover decomposition, $\{w_e^* \mid e \in \mathcal{E}_{bp}\}$ is an optimal solution to the edge-cover LP.

Given $\{w_e^* \mid e \in \mathcal{E}_{bp}\}$, we can apply the complementary slackness theorem (see Appendix A) to obtain an optimal solution $\{\nu'_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ to the vertex-pack LP satisfying:

- $\nu'_X + \nu'_Y = 1$ for each edge $(X, Y) \in \Sigma^*$ (because $w^*_{(X,Y)} > 0$);
- $\nu'_X = 0$ for each center vertex $X$ of a directed star in $\Sigma^*$ having at least two edges (because $\sum_{X \in e} w^*_e > 1$);

It follows from the above that $\nu'_X = 1$ for each petal vertex $X$ of a directed star in $\Sigma^*$ having at least two edges.

Now we construct the desired solution $\{\nu^*_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ to the vertex-pack LP:

- $\nu^*_X = \nu'_X$ for each vertex of a directed star in $\Sigma^*$ having at least two edges;
- For each directed star in $\Sigma^*$ with only one edge $(X, Y)$, if $\nu'_X \leq 0.5$, set $\nu^*_X = 0$ and $\nu^*_Y = 1$; otherwise, set $\nu^*_Y = 0$ and $\nu^*_X = 1$.

It is obvious that the set $\{\nu^*_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ thus constructed satisfies the three conditions stated in Lemma E.1. To prove the lemma, it remains to show that $\{\nu^*_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ is an optimal solution to the vertex-pack LP.

By our construction, $\nu^*_X \geq 0$ for each $X \in \mathcal{S} \cup \mathcal{T}$. Next, we argue that $\nu^*_X + \nu^*_Y \leq 1$ for each edge $(X, Y) \in \mathcal{E}_{bp}$. Assume that there exists an edge $(X, Y) \in \mathcal{E}_{bp}$ with $\nu^*_X + \nu^*_Y > 1$, meaning $\nu^*_X = \nu^*_Y = 1$ (because $\nu^*_X$ and $\nu^*_Y$ are either 0 or 1). As $\nu^*_X = 1$, one of the following must apply:

- $X$ is a petal vertex of a directed star in $\Sigma^*$ having at least two edges; in this case, $\nu'_X = 1$.
- $X$ is a vertex of a directed star in $\Sigma^*$ having only one edge; in this case, $\nu'_X > 0.5$.

Similarly, as $\nu^*_Y = 1$, one of the following must apply:

- $Y$ is a petal vertex of a directed star in $\Sigma^*$ having at least two edges; in this case, $\nu'_Y = 1$.
- $Y$ is a vertex of a directed star in $\Sigma^*$ having only one edge; in this case, $\nu'_Y \geq 0.5$.

Hence, $\nu'_X + \nu'_Y$ must be strictly greater than 1, which, however, contradicts the fact that $\{\nu'_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ is a solution to vertex-pack LP.

We can now assert that $\{\nu^*_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ is an optimal solution to the vertex-pack LP because $\sum_{X \in \mathcal{S} \cup \mathcal{T}} \nu^*_X = \sum_{X \in \mathcal{S} \cup \mathcal{T}} \nu'_X$ and $\{\nu'_X \mid X \in \mathcal{S} \cup \mathcal{T}\}$ is an optimal solution to the vertex-pack LP.

# References


[1] A. Abboud, S. Khoury, O. Leibowitz, and R. Safier. Listing 4-cycles. *CoRR*, abs/2211.10022, 2022.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *SIGMOD*, pages 275–286, 1999.

[4] N. 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] S. Assadi, M. Kapralov, and S. Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *ITCS*, pages 6:1–6:20, 2019.

[6] A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. *SIAM J. Comput.*, 42(4):1737–1767, 2013.

[7] M. Bentert, T. Fluschnik, A. Nichterlein, and R. Niedermeier. Parameterized aspects of triangle enumeration. *JCSS*, 103:61–77, 2019.

[8] A. Bjorklund, R. Pagh, V. V. Williams, and U. Zwick. Listing triangles. In *ICALP*, pages 223–234, 2014.

[9] S. Chaudhuri, R. Motwani, and V. R. Narasayya. On random sampling over joins. In *SIGMOD*, pages 263–274, 1999.

[10] Y. Chen and K. Yi. Random sampling and size estimation over cyclic joins. In *ICDT*, pages 7:1–7:18, 2020.

[11] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. of Comp.*, 14(1):210–223, 1985.

[12] K. Deeds, D. Suciu, M. Balazinska, and W. Cai. Degree sequence bound for join cardinality estimation. In *ICDT*, volume 255, pages 8:1–8:18, 2023.

[13] S. Deng, S. Lu, and Y. Tao. On join sampling and the hardness of combinatorial output-sensitive join algorithms. In *PODS*, pages 99–111, 2023.

[14] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms Appl.*, 3(3):1–27, 1999.

[15] H. Fichtenberger, M. Gao, and P. Peng. Sampling arbitrary subgraphs exactly uniformly in sublinear time. In *ICALP*, pages 45:1–45:13, 2020.

[16] T. Gogacz and S. Torunczyk. Entropy bounds for conjunctive queries with functional dependencies. In *ICDT*, volume 68, pages 15:1–15:17, 2017.

[17] C. T. Hoang, M. Kaminski, J. Sawada, and R. Sritharan. Finding and listing induced paths and cycles. *Discrete Applied Mathematics*, 161(4-5):633–641, 2013.

[18] S. V. M. Jayaraman, C. Ropell, and A. Rudra. Worst-case optimal binary join algorithms under general $\ell_p$ constraints. *CoRR*, abs/2112.01003, 2021.

[19] C. Jin and Y. Xu. Removing additive structure in 3sum-based reductions. In *STOC*, pages 405–418, 2023.

[20] M. Joglekar and C. Re. It's all a matter of degree - using degree information to optimize multiway joins. *Theory Comput. Syst.*, 62(4):810–853, 2018.

[21] M. A. Khamis, V. Nakos, D. Olteanu, and D. Suciu. Join size bounds using lp-norms on degree sequences. *CoRR*, abs/2306.14075, 2023.

[22] M. A. Khamis, H. Q. Ngo, and D. Suciu. Computing join queries with functional dependencies. In *PODS*, pages 327–342, 2016.

[23] M. A. Khamis, H. Q. Ngo, and D. Suciu. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? In *PODS*, pages 429–444, 2017.

[24] K. Kim, J. Ha, G. Fletcher, and W. Han. Guaranteeing the $\tilde{O}$(AGM/OUT) runtime for uniform sampling and size estimation over joins. In *PODS*, pages 113–125, 2023.

[25] V. Leis, A. Gubichev, A. Mirchev, A. K. Peter Boncz, and T. Neumann. How good are query optimizers, really? *PVLDB*, 9(3):204–215, 2015.

[26] G. Manoussakis. Listing all fixed-length simple cycles in sparse graphs in optimal time. In *Fundamentals of Computation Theory*, pages 355–366, 2017.

[27] J. Matousek and B. Gartner. *Understanding and Using Linear Programming*. Springer, 2007.

[28] G. Navarro, J. L. Reutter, and J. Rojas-Ledesma. Optimal joins using compact data structures. In *ICDT*, volume 155, pages 21:1–21:21, 2020.

[29] J. Nesetril and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[30] H. Q. Ngo. Worst-case optimal join algorithms: Techniques, results, and open problems. In *PODS*, pages 111–124, 2018.

[31] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. Worst-Case Optimal Join Algorithms: [Extended Abstract]. In *PODS*, pages 37–48, 2012.

[32] H. Q. Ngo, E. Porat, C. Re, and A. Rudra. Worst-case optimal join algorithms. *JACM*, 65(3):16:1–16:40, 2018.

[33] H. Q. Ngo, C. Re, and A. Rudra. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Rec.*, 42(4):5–16, 2013.

[34] R. L. T. Santos, C. MacDonald, and I. Ounis. Search result diversification. *Found. Trends Inf. Retr.*, 9(1):1–90, 2015.

[35] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.

[36] R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[37] D. Suciu. Applications of information inequalities to database theory problems. *CoRR*, abs/2304.11996, 2023.

[38] M. M. Syslo. An efficient cycle vector space algorithm for listing all cycles of a planar graph. *SIAM J. of Comp.*, 10(4):797–808, 1981.

[39] T. L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In *ICDT*, pages 96–106, 2014.

[40] Z. Zhao, R. Christensen, F. Li, X. Hu, and K. Yi. Random sampling over joins revisited. In *SIGMOD*, pages 1525–1539, 2018.