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

Ru Wang Yufei Tao

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

December 20, 2023

Abstract

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

We then utilize our techniques to tackle *directed subgraph sampling*, a problem that has extensive database applications and bears close relevance to joins. Let $G = (V, E)$ be a directed data graph where each vertex has an out-degree at most λ, and let P be a directed pattern graph with a constant number of vertices. The objective is to uniformly sample an occurrence of P in G. The problem can be modeled as join sampling with input size $\mathrm{IN} = \Theta(|E|)$ but, whenever P contains cycles, the converted join has *cyclic* degree constraints. We show that it is always possible to throw away certain degree constraints such that (i) the remaining constraints are acyclic and (ii) the new join has asymptotically the same polymatroid bound *polymat* as the old one. Combining this finding with our new join sampling solution yields an algorithm to sample from the original (cyclic) join (thereby yielding a uniformly random occurrence of P) in $O(\mathit{polymat}/\max\{1,\mathrm{OUT}\})$ expected time after $O(|E|)$ expected-time preprocessing, where OUT is the number of occurrences. We also prove similar results for *undirected subgraph sampling* and demonstrate how our techniques can be significantly simplified in that scenario. Previously, the state of the art for (undirected/directed) subgraph sampling uses $O(|E|^{\rho^*}/\max\{1,\mathrm{OUT}\})$ time to draw a sample (after $O(|E|)$ expected-time preprocessing) where ρ^* is the fractional edge cover number of P. Our results are more favorable because *polymat* never exceeds but can be considerably lower than $|E|^{\rho^*}$.

1 Introduction

In relational database systems, (natural) joins are acknowledged as notably computation-intensive, with its cost surging drastically in response to expanding data volumes. In the current big data era, the imperative to circumvent excessive computation increasingly overshadows the requirement for complete join results. A myriad of applications, including machine learning algorithms, online analytical processing, and recommendation systems, can operate effectively with random samples. This situation has sparked research initiatives focused on devising techniques capable of producing samples from a join result significantly faster than executing the join in its entirety. In the realm of graph theory, the significance of join operations is mirrored in their intrinsic connections to *subgraph listing*, a classical problem that seeks to pinpoint all the occurrences of a pattern P (for instance, a directed 3-vertex cycle) within a data graph G (such as a social network where a directed edge symbolizes a "follow" relationship). Analogous to joins, subgraph listing demands a vast amount of computation time, which escalates rapidly with the sizes of G and P. Fortunately, many social network analyses do not require the full set of occurrences of P, but can function well with only samples from those occurrences. This has triggered the development of methods that can extract samples considerably faster than finding all the occurrences.

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

1.1 Problem Definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Undirected Graph Sampling. The setup of this problem is the same as the previous problem except that (i) both the data graph $G = (V, E)$ and the pattern graph $P = (V_P, E_P)$ are simple undirected graphs, with P being connected; (ii) a subgraph G_{sub} of G is an *occurrence* of P if G_{sub} and P are isomorphic in the undirected sense: there is a *isomorphism bijection* $f : V_{sub} \to V_P$ between P and G_{sub} such that, for any distinct $u_1, u_2 \in V_{sub}$, an edge $\{u_1, u_2\}$ exists in E_{sub} if and only if $\{f(u_1), f(u_2)\} \in E_P$;[2] and (iii) the degree constraint becomes: every vertex in G has a

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

[2]We represent a directed edge as an ordered pair and an undirected edge as a set.

degree at most λ.

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

1.2 Related Work

Join Computation. Any algorithm correctly answering a join query $\mathcal{Q}$ must incur $\Omega(\text{OUT})$ time just to output the OUT tuples in $join(\mathcal{Q})$. 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* [7]. 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 $X \in \mathcal{V}$, it holds that $\sum_{F \in \mathcal{E}: X \in F} w(F) \geq 1$ (namely, the total weight assigned to the hyperedges covering X is at least 1). Atserias, Grohe, and Marx [7] showed that, given any fractional edge cover, it always holds that $\text{OUT} \leq \prod_{F \in \mathcal{E}} |R_F|^{w(F)}$, where R_F is the relation in $\mathcal{Q}$ whose schema corresponds to the hyperedge F. The AGM bound is defined as $AGM(\mathcal{Q}) = \min_w \prod_{F \in \mathcal{E}} |R_F|^{w(F)}$.

The AGM bound is tight: given any hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and any set of positive integers $\{N_F \mid F \in \mathcal{E}\}$, there is always a join $\mathcal{Q}$ such that $\mathcal{Q}$ has $\mathcal{G}$ as the schema graph, $|R_F| = |N_F|$ for each $F \in \mathcal{E}$, and the output size OUT is $\Theta(AGM(\mathcal{Q}))$. This has motivated the development of algorithms [5, 14, 21, 23, 25, 28, 31–34, 38] 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 [13, 17, 21, 22, 24, 25, 30, 36] in establishing refined upper bounds on OUT tailored for more complex degree constraints. Most notably, Khamis et al. [25] proposed the *entropic bound*, which is applicable to any set DC of degree constraints and is tight in a strong sense (see Theorem 5.5 of [36]). 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. [25] introduced the *polymatroid bound* as an alternative, which we represent as $polymat(\mathsf{DC})$ because this bound is fully decided by DC (i.e., any join $\mathcal{Q} \models \mathsf{DC}$ must satisfy $\text{OUT} \leq polymat(\mathsf{DC})$). Section 2 will discuss $polymat(\mathsf{DC})$ in detail; for now, it suffices to understand that (i) the polymatroid bound, although possibly looser than the entropic bound, never exceeds the AGM bound, and (ii) $polymat(\mathsf{DC})$ can be computed in $O(1)$ time under data complexity. Khamis et al. [25] proposed an algorithm named PANDA that can evaluate an arbitrary join $\mathcal{Q} \models \mathsf{DC}$ in time $\tilde{O}(polymat(\mathsf{DC}))$.

Interestingly, when DC is acyclic, the entropic bound is equivalent to the polymatroid bound [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 [39]. The problem becomes more complex when dealing with an arbitrary (cyclic) join $\mathcal{Q}$, with the latest advancements presented in two PODS'23 papers [14, 26]. Specifically, Kim et al. [26] 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. [14] 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, 10, 11, 14, 26, 39] and the references therein for other results (now superseded) on join sampling.

Subgraph Listing. Let us start by clarifying the *fractional edge cover number* $\rho^*(P)$ of a simple undirected pattern graph $P = (V_P, E_P)$. Given a fractional edge cover of P (i.e., function $w : E_P \to [0, 1]$ such that, for any vertex $X \in V_P$, we have $\sum_{F \in E_P : X \in F} w(F) \geq 1$), define $\sum_{F \in E_P} w(F)$ as the *total weight* of w. The value of $\rho^*(P)$ is the smallest total weight of all fractional edge covers of P. Given a directed pattern graph P, we define its fractional edge cover number $\rho^*(P)$ as the value $\rho^*(P')$ of the corresponding undirected graph P' that is obtained from P by ignoring all the edge directions.

When P has a constant size, it is well-known [4, 7] that any data graph $G = (V, E)$ can encompass $O(|E|^{\rho^*(P)})$ occurrences of P. This holds true both when P and G are directed and when they are undirected. This upper bound is tight: in both the directed and undirected scenarios, for any integer m, there is a data graph $G = (V, E)$ with $|E| = m$ edges that has $\Omega(m^{\rho^*(P)})$ occurrences of P. Thus, a subgraph listing algorithm is considered worst-case optimal if it finishes in $\tilde{O}(|E|^{\rho^*(P)})$ time.

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

Assuming P and G to be directed, Jayaraman et al. [19] presented interesting enhancement over the above transformation in the scenario where each vertex of G has an out-degree at most λ. The key lies in examining the polymatroid bound of the join $\mathcal{Q}$ derived from subgraph listing. As will be explained in Section 4, this join $\mathcal{Q}$ has a set DC of degree constraints whose constraint dependency graph G_{DC} coincides with P. Jayaraman et al. developed an algorithm that lists all occurrences of $\mathcal{Q}$ in G in $O(\textit{polymat}(\mathsf{DC}))$ time (after a preprocessing of $O(\text{IN})$ expected time) and confirmed that this is worst-case optimal. Their findings are closely related to our work, and we will delve into them further when their specifics become crucial to our discussion.

There is a substantial body of literature on bounding the cost of subgraph listing using parameters distinct from those already mentioned. These studies typically concentrate on specific patterns (such as paths, cycles, and cliques) or particular graphs (for instance, those that are sparse under a suitable metric). We refer interested readers to [1, 8, 9, 12, 15, 18, 20, 27, 29, 37] and the references therein.

Subgraph Sampling. Fichtenberger, Gao, and Peng [16] 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 [14], Deng et al. clarified how to deploy an arbitrary join sampling algorithm to perform subgraph sampling; their approach ensures the same guarantees as in [16], baring an $\tilde{O}(1)$ factor. The methods of [14, 16] are applicable in both undirected and directed scenarios.

1.3 Our Results

For any join $\mathcal{Q}$ with an acyclic set DC of degree constraints, we will demonstrate in Section 3 how to extract a uniformly random sample from $join(\mathcal{Q})$ in $O(polymat(\mathsf{DC})/\max\{1, \mathrm{OUT}\})$ expected time, following an initial preprocessing of $O(\mathrm{IN})$ expected time. This performance is favorable when compared to the recent results of [14, 26] (reviewed in Section 1.2), which examined settings where DC consists only of cardinality constraints and is therefore trivially acyclic. As $polymat(\mathsf{DC})$ is at most but can be substantially lower than $AGM(\mathcal{Q})$, our guarantees are never worse, but can be considerably better, than those in [14, 26].

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

Our next contribution is to prove that, interestingly, the issue does not affect subgraph listing/sampling. Consider first directed subgraph listing, defined by a pattern graph P and a data graph G where every vertex has an out-degree at most λ. This problem can be converted to a join $\mathcal{Q}$ on binary relations, which is associated with a set DC of degree constraints such that the constraint dependency graph G_{DC} is exactly P. Consequently, whenever P contains a cycle, so does G_{DC}, making DC cyclic. Nevertheless, we will demonstrate in Section 4 the existence of an acyclic set $\mathsf{DC}' \subset \mathsf{DC}$ ensuring $Q \models \mathsf{DC}'$ and $polymat(\mathsf{DC}) = \Theta(polymat(\mathsf{DC}'))$. This "magical" DC' has an immediate implication: Ngo's join algorithm in [30], when applied to Q and DC' directly, already solves directed subgraph listing optimally in $O(polymat(\mathsf{DC}')) = O(polymat(\mathsf{DC}))$ time. This dramatically simplifies — in terms of both procedure and analysis — an algorithm of Jayaraman et al. [19] (for directed subgraph listing, reviewed in Section 1.2) that has the same guarantees.

The same elegance extends to directed subgraph sampling: by applying our new join sampling algorithm to $\mathcal{Q}$ and the "magical" DC', we can sample an occurrence of P in G using $O(polymat(\mathsf{DC})/\max\{1, \mathrm{OUT}\})$ expected time, after a preprocessing of $O(|E|)$ expected time. As $polymat(\mathsf{DC})$ 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 [14, 16, 26] reviewed in Section 1.2.

Undirected subgraph sampling (where both P and G are undirected and each vertex in G has a degree at most λ) is a special version of its directed counterpart and can be settled by a slightly modified version of our directed subgraph sampling algorithm. However, it is possible to do better by harnessing the undirected nature. In Section 5, we will first solve the polymatroid bound into a *closed-form* expression, which somewhat unexpectedly exhibits a crucial relationship to a well-known graph decomposition method. This relationship motivates a surprisingly simple algorithm for undirected subgraph sampling that offers guarantees analogous to those in the directed scenario.

By virtue of the power of sampling, our findings have further implications on other fundamental problems including output-size estimation, output permutation, and small-delay enumeration. We will elaborate on the details in Section 6.

2 Preliminaries

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

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

Define:

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

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

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

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

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

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

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

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

Join Output Size Bounds. Let us fix a join $\mathcal{Q}$ whose schema graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Suppose that $\mathcal{Q}$ is consistent with a set DC of degree constraints, i.e., $\mathcal{Q} \models \mathsf{DC}$. As explained in Section 1.1, we follow the convention that each relation of $\mathcal{Q}$ implicitly inserts a cardinality constraint (i.e., a special degree constraint) to DC. Note that the set DC is merely a rule collection over $\mathcal{V}$. The following lemma was established by Khamis et al. [25]:

Lemma 2.1 ([25]). *The output size* OUT *of $\mathcal{Q}$ is at most polymat*(DC)*, i.e., the polymatroid bound of* DC *(as defined in* (4)*).*

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

Lemma 2.2 ([30]). *When* DC *is acyclic, it always holds that modular*(DC) $=$ *polymat*(DC)*, namely,* $\max_{h \in \Gamma_{\mathcal{V}} \cap \mathcal{H}_{\mathsf{DC}}} h(\mathcal{V}) = \max_{h \in \mathsf{M}_{\mathcal{V}} \cap \mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})$.

As a corollary, when DC is acyclic, the value of *modular*(DC) always serves as an upper bound of OUT. In our technical development, we will need to analyze the set functions $h^* \in \Gamma_{\mathcal{V}}$ that realize the polymatriod bound, i.e., $h^*(\mathcal{V}) = \max_{h \in \Gamma_{\mathcal{V}} \cap \mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})$. A crucial advantage provided by Lemma 2.2 is that we can instead scrutinize those set functions $h^* \in \mathsf{M}_{\mathcal{V}}$ realizing the *modular* bound, i.e., $h^*(\mathcal{V}) = \max_{h \in \mathsf{M}_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$. Compared to their submodular counterparts, modular set functions exhibit more regularity because every $h \in \mathsf{M}_\mathcal{V}$ is fully determined by its value $h(\{A\})$ on each *individual* attribute $A \in \mathcal{V}$. In particular, for any $h \in \mathsf{M}_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}$, it holds true that $h(\mathcal{Y}) - h(\mathcal{X}) = \sum_{A \in \mathcal{Y} - \mathcal{X}} h(A)$ for any $\mathcal{X} \subset \mathcal{Y} \subseteq \mathcal{V}$. If we associate each $A \in \mathcal{V}$ with a variable ν_A, then $\max_{h \in \mathsf{M}_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$ — hence, also $\max_{h \in \Gamma_\mathcal{V} \cap \mathcal{H}_\mathsf{DC}} h(\mathcal{V})$ — is precisely the optimal value of the following LP:

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

$$\begin{aligned} &\sum_{A \in \mathcal{Y} - \mathcal{X}} \nu_A \le \log N_{\mathcal{Y}|\mathcal{X}} && \forall (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC} \\ &\nu_A \ge 0 && \forall A \in \mathcal{V} \end{aligned}$$

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 $\min \sum_{(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} \delta_{\mathcal{Y}|\mathcal{X}} \cdot \log N_{\mathcal{Y}|\mathcal{X}}$ subject to

$$\begin{aligned} &\sum_{\substack{(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC} \\ A \in \mathcal{Y} - \mathcal{X}}} \delta_{\mathcal{Y}|\mathcal{X}} \ge 1 && \forall A \in \mathcal{V} \\ &\delta_{\mathcal{Y}|\mathcal{X}} \ge 0 && \forall (\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC} \end{aligned}$$

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 $O(\text{IN})$ expected time a data structure that supports each join sampling operation in $O(\textit{polymat}(\mathsf{DC}) / \max\{1, \text{OUT}\})$ expected time, where IN and OUT are the input and out sizes of $\mathcal{Q}$, respectively, and $\textit{polymat}(\mathsf{DC})$ is the polymatroid bound of DC.*

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

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

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

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

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

where σ and $\ltimes$ are the standard selection and semi-join operators in relational algebra, respectively. To understand the intuition behind $\textit{reldeg}_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a)$, imagine drawing a tuple $\boldsymbol{u}$ from $\Pi_\mathcal{Y}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes$

```
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.     $a_i \leftarrow \boldsymbol{u}^\circ(A_i)$
6.     if $(\mathcal{X}^\circ, \mathcal{Y}^\circ, N_{\mathcal{Y}^\circ|\mathcal{X}^\circ}) \neq constraint^*_{i-1}(\boldsymbol{w}_{i-1}, a_i)$ then declare failure
7.     $\boldsymbol{w}_i \leftarrow$ the tuple over $\mathcal{V}_i$ formed by concatenating $\boldsymbol{w}_{i-1}$ with $a_i$
8.     declare failure with probability $1 - p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i)$, where $p_{\text{pass}}$ is given in (12)
9. if $\boldsymbol{w}_k[F] \in R_F$ for $\forall F \in \mathcal{E}$ then          /* that is, $\boldsymbol{w}_k \in join(\mathcal{Q})$ */
10.    return $\boldsymbol{w}_k$
```

Figure 1: Our sampling algorithm

$\boldsymbol{w}$) uniformly at random; then $reldeg_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a)$ is the probability to see $\boldsymbol{u}(A_i) = a$. Given a tuple $\boldsymbol{w}$ over $\mathcal{V}_{i-1}$ and a value $a \in \mathbf{dom}$, define

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

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

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

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

$$\begin{aligned}
\prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} N_{\mathcal{Y}|\mathcal{X}}^{\delta_{\mathcal{Y}|\mathcal{X}}} &= \exp_2\Big(\sum_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} \delta_{\mathcal{Y}|\mathcal{X}} \cdot \log N_{\mathcal{Y}|\mathcal{X}}\Big) = \exp_2\Big(\max_{h \in \mathsf{M}_{\mathcal{V}} \cap \mathcal{H}_{\mathsf{DC}}} h(\mathcal{V})\Big) \\
\text{(by (5))} &= modular(\mathsf{DC}) \\
\text{(by Lemma 2.2)} &= polymat(\mathsf{DC}).
\end{aligned} \tag{10}$$

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

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

Two observations will be useful later:

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

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

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

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

or otherwise terminate the algorithm by declaring failure. As proved later, $p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i)$ cannot exceed 1 (Lemma 3.2); moreover, this value can be computed in $O(1)$ time. The overall execution time of $\mathsf{ADC\text{-}sample}$ is constant.

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

Lemma 3.2. *For every $i \in [k]$, we have $p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i) \le 1$.*

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

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

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

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

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

We can now derive

$$p_{\text{pass}}(i, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i) \quad = \quad \frac{1}{reldeg^*_i(\boldsymbol{w}_{i-1}, a_i)} \prod_{(\mathcal{X},\mathcal{Y},N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}} \Big(\frac{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_i)}{deg_{\mathcal{Y}|\mathcal{X}}(R_{F(\mathcal{X},\mathcal{Y})} \ltimes \boldsymbol{w}_{i-1})}\Big)^{\delta_{\mathcal{Y}|\mathcal{X}}}$$

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

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

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

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

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

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

Observe

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

On the other hand:

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

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

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

$$= \frac{1}{B_0(\text{null})} \cdot \prod_{i=1}^{k} \frac{1}{|\mathsf{DC}(A_i)|}.$$

As the above is identical for every $\boldsymbol{v} \in join(\mathcal{Q})$, 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 $join(\mathcal{Q})$ with probability

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

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

Performing a Join Sampling Operation. Recall that this operation must either return a uniformly random sample of $join(\mathcal{Q})$ or declare $join(\mathcal{Q}) = \emptyset$. To support this operation, we execute two *threads* concurrently. The first thread repeatedly invokes ADC-sample until it successfully returns a sample. The other thread runs Ngo's algorithm in [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.

This strategy guarantees that the join operation completes in $O(polymat(\mathsf{DC})/\max\{1, \text{OUT}\})$ time. To explain why, consider first the scenario where $\text{OUT} \geq 1$. In this case, we expect to find a sample with $O(polymat(\mathsf{DC})/\text{OUT})$ repeats of ADC-sample. Hence, the first thread finishes in $O(polymat(\mathsf{DC})/\text{OUT})$ expected sample time. On the other hand, if $\text{OUT} = 0$, 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. [26]. In retrospect, two main obstacles prevent an obvious extension of their algorithm to an arbitrary acyclic DC. The first is identifying an appropriate way to deal with constraints $(\mathcal{X}, \mathcal{Y}, N_{\mathcal{Y}|\mathcal{X}}) \in \mathsf{DC}$ where $\mathcal{X} \neq \emptyset$ (such constraints are absent in the degenerated context of [26]). The second obstacle involves determining how to benefit from a topological order (attribute ordering is irrelevant in [26]); replacing the order with a non-topological one may ruin the correctness of ADC-sample.

4 Directed Subgraph Sampling

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

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

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

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

This formulation reflects how directed subgraph listing can be processed as a join. Consider a *companion join* $\mathcal{Q}$ constructed from G and P as follows. The schema graph of $\mathcal{Q}$, denoted as

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

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

When P is acyclic, so is G_{DC}, and thus our algorithm in Theorem 3.1 can be readily applied to handle a subgraph sampling operation. To analyze the performance, consider first $\mathrm{OUT} \geq 1$. We expect to draw $O(\mathrm{OUT}_{\mathcal{Q}}/\mathrm{OUT})$ samples from $\mathit{join}(\mathcal{Q})$ until a success event. As Theorem 3.1 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. We run a concurrent thread that executes Ngo's algorithm in [30], which finds the whole $\mathit{join}(\mathcal{Q})$, and hence $\mathit{occ}(G, P)$, in $O(\mathit{polymat}(\mathsf{DC}))$ time, after which we can declare $\mathit{occ}(G, P) = \emptyset$ or sample from $\mathit{occ}(G, P)$ in $O(1)$ time. By accepting whichever thread finishes earlier, we ensure that the operation completes in $O(\mathit{polymat}(\mathsf{DC})/\max\{1, \mathrm{OUT}\})$ time.

The main challenge arises when P is cyclic. In this case, G_{DC} (which equals P) is cyclic. Thus, DC becomes a cyclic set of degree constraints, rendering neither Theorem 3.1 nor Ngo's algorithm in [30] applicable. We overcome this challenge with the lemma below.

Lemma 4.1. *If DC is cyclic, we can always find an acyclic subset $\mathsf{DC}' \subset \mathsf{DC}$ satisfying $\mathit{polymat}(\mathsf{DC}') = \Theta(\mathit{polymat}(\mathsf{DC}))$.*

The proof is presented in Appendix B. Because $\mathcal{Q} \models \mathsf{DC}$ and DC' is a subset of DC, we know that $\mathcal{Q}$ must be consistent with DC' as well, i.e., $\mathcal{Q} \models \mathsf{DC}'$. Therefore, our Theorem 3.1 can now be used to extract a sample from $\mathit{join}(\mathcal{Q})$ in $O(\mathit{polymat}_{dir}(\mathsf{DC}')/\max\{1, \mathrm{OUT}_{\mathcal{Q}}\})$ time. Importantly, Lemma 4.1 also permits us to directly apply Ngo's algorithm in [30] to compute $\mathit{join}(\mathcal{Q})$ in $O(\mathit{polymat}(\mathsf{DC}'))$ time. Therefore, we can now apply the two-thread technique to sample from $\mathit{occ}(G, P)$ in

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

time. We thus have arrived yet:

Theorem 4.2. *Let $G = (V, E)$ be a simple directed data graph, where each vertex has an out-degree at most λ. Let $P = (V_P, E_P)$ be a simple weakly-connected directed pattern graph with a constant* *number of vertices. We can build in $O(|E|)$ expected time a data structure that supports each subgraph sampling operation in $O(polymat_{dir}(|E|, \lambda, P)/\max\{1, \mathrm{OUT}\})$ expected time, where* OUT *is the number of occurrences of P in G, and $polymat_{dir}(|E|, \lambda, P)$ is the directed polymatrioid bound in* (18).

Remarks. For subgraph *listing*, Jayaraman et al. [19] presented a sophisticated method that also enables the application of Ngo's algorithm in [30] to a cyclic P. Given the companion join $\mathcal{Q}$, they employ the *degree uniformization technique* [21] 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 DC_i of degree constraints (which is not always a subset of DC) with the property $\sum_{i=1}^{t} polymat(\mathsf{DC}_i) \leq polymat(\mathsf{DC})$. Each join $\mathcal{Q}_i$ ($i \in [t]$) can then be processed by Ngo's algorithm in $O(polymat(\mathsf{DC}_i))$ time, thus giving an algorithm for computing $join(\mathcal{Q})$ (and hence $occ(G, P)$) in $O(polymat(\mathsf{DC}))$ time. On the other hand, Lemma 4.1 facilitates a *direct* application of Ngo's algorithm to $\mathcal{Q}$, implying the non-necessity of degree uniformization in subgraph listing. We believe that this simplification is noteworthy and merits its own dedicated exposition, considering the critical nature of the subgraph listing problem. In the absence of Lemma 4.1, integrating our join-sampling algorithm with the methodology of [19] for the purpose of subgraph sampling would require substantially more effort. Our proof of Lemma 4.1 *does* draw upon the analysis of [19], as discussed in depth in Appendix B.

5 Undirected Subgraph Sampling

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

We formulate the "polymatroid bound" of this problem through a reduction to its directed counterpart. For $P = (V_P, E_P)$, create a *directed* pattern $P' = (V'_P, E'_P)$ as follows. First, set $V'_P = V_P$. Second, for every edge $\{X, Y\} \in E_P$, add to E_P two directed edges (X, Y) and (Y, X). Now, given an integer $m \geq 1$, an integer $\lambda \in [1, m]$, and an undirected pattern $P = (V_P, E_P)$, the *undirected polymatroid bound* of m, λ, and P is defined as

$$polymat_{undir}(m, \lambda, P) \quad = \quad polymat_{dir}(m, \lambda, P') \tag{19}$$

where the function $polymat_{dir}$ is defined in (18).

Our formulation highlights how undirected subgraph sampling can be reduced to the directed version. From $G = (V, E)$, construct a *directed* graph $G' = (V', E')$ by setting $V' = V$, and for every edge $\{x, y\} \in E$, adding to E' two directed edges (x, y) and (y, x). Every occurrence in $occ(G, P)$ matches the same number (a constant) of occurrences in $occ(G', P')$. By resorting to Theorem 4.2, we can obtain an algorithm to sample from $occ(G, P)$ in $O(polymat_{undir}(2|E|, \lambda, P)/\max\{1, \mathrm{OUT}\})$ time (where $\mathrm{OUT} = |occ(G, P)|$) after a preprocessing of $O(|E|)$ expected time. We omit the details because they will be superseded by another simpler approach to be described later.

Unlike $polymat_{dir}(m, \lambda, P)$, which is defined through an LP, we can solve the undirected counterpart $polymat_{undir}(m, \lambda, P)$ into a closed form. It is always possible [6,33,35] to decompose P into *vertex-disjoint* subgraphs $\pentagon_1, \pentagon_2, ..., \pentagon_\alpha$, $\star_1, \star_2, ...$, and $\star_\beta$ (for some integers $\alpha, \beta \geq 0$) such that

- $\pentagon_i$ is an odd-length cycle for each $i \in [\alpha]$;
- $\star_j$ is a star[3] for each $j \in [\beta]$;

[3]A *star* is an undirected graph where one vertex, called the *center*, has an edge to every other vertex, and each non-center vertex has degree 1.

- $\sum_{i=1}^{\alpha} \rho^*(\pentagon_i) + \sum_{j=1}^{\beta} \rho^*(\star_j) = \rho^*(P)$; see Section 1.2 for the definition of the fractional edge cover number function $\rho^*(.)$.

We will refer to $(\pentagon_1, ..., \pentagon_\alpha, \star_1, ..., \star_\beta)$ as a *fractional edge-cover decomposition* of P. Define:

$$k_{cycle} = \text{total number of vertices in } \pentagon_1, ..., \pentagon_\alpha \tag{20}$$
$$k_{star} = \text{total number of vertices in } \star_1, ..., \star_\beta. \tag{21}$$

We establish the lemma below in Appendix C.

Lemma 5.1. *If k is the number of vertices in P, then*

$$polymat_{undir}(m, \lambda, P) = \begin{cases} m \cdot \lambda^{k-2} & \text{if } \lambda \le \sqrt{m}, \\ m^{\frac{k_{cycle}}{2}+\beta} \cdot \lambda^{k_{star}-2\beta} & \text{if } \lambda > \sqrt{m} \end{cases} \tag{22}$$

As shown in Appendix D, for $k = O(1)$, the expression in (22) asymptotically matches an upper bound of $|occ(G', P)|$ — for any G' with m edges and maximum vertex-degree at most λ — easily derived from a fractional edge-cover decomposition. The same appendix will also prove that, for a wide range of m and λ values, there exists a graph G' with m edges and maximum vertex degree at most λ guaranteeing the following *simultaneously* for all patterns P with k vertices: $|occ(G', P)|$ is no less than the expression's value, up to a factor of $1/(4k)^k$. This implies that, when k is a constant, the expression *asymptotically* equals $polymat_{undir}(m, \lambda, P)$. More effort is required to prove that the expression equals $polymat_{undir}(m, \lambda, P))$ *precisely*, as we demonstrate in Appendix C.

Lemma 5.1 points to an alternative strategy to perform undirected subgraph sampling without joins. Identify an arbitrary spanning tree $\mathcal{T}$ of the pattern $P = (V_P, E_P)$. Order the vertices of V_P as $A_1, A_2, ..., A_k$ such that, for any $i \in [2, k]$, vertex A_i is adjacent in $\mathcal{T}$ to a (unique) vertex A_j with $j \in [i-1]$. Now construct a map $f : V_P \to V$ as follows (recall that V is the vertex set of the data graph $G = (V, E)$). First, take an edge $\{u, v\}$ uniformly at random from E, and choose one of the following with an equal probability: (i) $f(A_1) = u, f(A_2) = v$, or (ii) $f(A_1) = v, f(A_2) = u$. Then, for every $i \in [3, k]$, decide $f(A_i)$ as follows. Suppose that A_i is adjacent to A_j for some (unique) $j \in [1, i-1]$. Toss a coin with probability $deg(f(A_j))/\lambda$, where $deg(f(A_j))$ is the degree of vertex $f(A_j)$ in G. If the coin comes up tails, declare failure and terminate. Otherwise, set $f(A_i)$ to a neighbor of $f(A_j)$ in G picked uniformly at random. After finalizing the map $f(.)$, check whether $\{f(A_i) \mid i \in [k]\}$ induces a subgraph of G isomorphic to P. If so, *accept* f and return this subgraph; otherwise, *reject* f. To guarantee a sample or declare $occ(G, P) = \emptyset$, apply the "two-thread approach" by (i) repeating the algorithm until acceptance and (ii) concurrently running an algorithm for computing the whole $occ(G, P)$ in $O(polymat_{undir}(|E|, \lambda, P))$ time[4]. As proved in Appendix E, this ensures the expected sample time $O(|E| \cdot \lambda^{k-2}/\max\{1, \text{OUT}\})$ after a preprocessing of $O(|E|)$ expected time.

The above algorithm suffices for the case $\lambda \le \sqrt{|E|}$. Consider now the case $\lambda > \sqrt{|E|}$. We will construct a map $f : V_P \to V$ according to a given fractional edge-cover decomposition $(\pentagon_1, ..., \pentagon_\alpha, \star_1, ..., \star_\beta)$. For each $i \in [\alpha]$, use the algorithm of [16] to uniformly sample an occurrence of $\pentagon_i$ — denoted as $G_{sub}(\pentagon_i)$ — in $O(|E|^{\rho^*(\pentagon_i)}/\max\{1, |occ(G, \pentagon_i)|\})$ expected time. Let $f_{\pentagon_i}$ be a map from the vertex set of $\pentagon_i$ to that of $G_{sub}(\pentagon_i)$, chosen uniformly at random from all the isomorphism bijections (defined in Section 1.1) between $\pentagon_i$ and $G_{sub}(\pentagon_i)$. For each $j \in [\beta]$, apply our earlier algorithm to uniformly sample an occurrence of $\star_j$ — denoted as $G_{sub}(\star_j)$ — in $O(|E| \cdot \lambda^{k_j}/\max\{1, |occ(G, \star_j)|\})$ expected time, where k_j is the number of vertices in $\star_j$. Let

[4]For this purpose, use Ngo's algorithm in [30] to find all occurrences in $occ(G', P')$ — see our earlier definitions of G' and P' — which is possible due to Lemma 4.1.

$f_{\star_j}$ be a map from the vertex set of $\star_j$ to that of $G_{sub}(\star_j)$, chosen uniformly at random from all the polymorphism bijections between $\star_j$ and $G_{sub}(\star_j)$. If any of $\pentagon_1, ..., \pentagon_\alpha$, $\star_1, ..., \star_\beta$ has no occurrences, declare $occ(G, P) = \emptyset$ and terminate. Otherwise, the functions in $\{f_{\pentagon_i} \mid i \in [\alpha]\}$ and $\{f_{\star_j} \mid j \in [\beta]\}$ together determine the map f we aim to build. Check whether $\{f(A) \mid A \in V\}$ induces a subgraph of G isomorphic to P. If so, *accept* f and return this subgraph; otherwise, *reject* f. Repeat until acceptance and concurrently run an algorithm for computing the whole $occ(G, P)$ in $O(polymat_{undir}(|E|, \lambda, P))$ time. As proved in Appendix E, this ensures the expected sample time $O(|E|^{\frac{k_{cycle}}{2}+\beta} \cdot \lambda^{k_{star}-2\beta} / \max\{1, \mathrm{OUT}\})$ after a preprocessing of $O(|E|)$ expected time.

We can now conclude with our last main result:

Theorem 5.2. *Let $G = (V, E)$ be a simple undirected data graph, where each vertex has a degree at most λ. Let $P = (V_P, E_P)$ be a simple connected 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 $O(polymat_{undir}(|E|, \lambda, P)/\max\{1, \mathrm{OUT}\})$ expected time, where* OUT *is the number of occurrences of P in G, and $polymat_{undir}(|E|, \lambda, P)$ is the undirected polymatrioid bound in* (22)*.*

6 Concluding Remarks

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

- By standard techniques [11, 14], we can estimate the output size OUT up to a relative error ϵ with high probability (i.e., at least $1 - 1/\mathrm{IN}^c$ for an arbitrarily large constant c) in time $\tilde{O}(\frac{1}{\epsilon^2} \frac{polymat(\mathsf{DC})}{\max\{1,\mathrm{OUT}\}})$ after a preprocessing of $O(\mathrm{IN})$ expected time.
- Employing a technique in [14], we can, with high probability, report all the tuples in $join(\mathcal{Q})$ with a delay of $\tilde{O}(\frac{polymat(\mathsf{DC})}{\max\{1,\mathrm{OUT}\}})$. In this context, 'delay' refers to the maximum interval between the reporting of two successive result tuples, assuming the presence of a placeholder tuple at the beginning and another at the end.
- In addition to the delay guarantee, our algorithm in the second bullet can, with high probability, report the tuples of $join(\mathcal{Q})$ in a random permutation. This means that each of the OUT! possible permutations has an equal probability of being the output.

All of the results presented above compare favorably with the current state of the art as presented in [14]. This is primarily due to the superiority of $polymat(\mathsf{DC})$ over $AGM(\mathcal{Q})$. In addition, our findings in the last two bullet points also complement Ngo's algorithm as described in [30] in a satisfying manner.

Appendix

A Implementing **ADC-Sample** with Indexes

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 $O(\text{IN})$ 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 only 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 $O(\text{IN})$ 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 $a \in \mathbf{dom}$, we can compute $reldeg_{i,\mathcal{X},\mathcal{Y}}(\boldsymbol{w}, a)$ defined in (7) 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 (7)), first obtain $\boldsymbol{w}_1 = \boldsymbol{w}[schema(R) \cap \mathcal{V}_{i-1}]$. 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 $|\sigma_{A_i=a}(\Pi_{\mathcal{Y}}(R \ltimes \boldsymbol{w}))|$ (the numerator of (7)), we can first obtain $\boldsymbol{w}_2$, which is a tuple over $schema(R) \cap \mathcal{V}_i$ that shares the values of $\boldsymbol{w}_1$ on all the attributes in $schema(R) \cap \mathcal{V}_{i-1}$ and additionally uses value a on attribute A_i. Then, $\sigma_{A_i=a}(\Pi_{\mathcal{Y}}(R \ltimes \boldsymbol{w}))$ 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 $a \in \mathbf{dom}$, we can compute $reldeg_i^*(\boldsymbol{w}, a)$ and $constraint_i^*(\boldsymbol{w}, a)$ — defined in (8) and (9), respectively — in constant time.

It remains to explain how to implement Line 4 of ADC-sample (Figure 1). 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 $\boldsymbol{w}' = \boldsymbol{w}_{i-1}[schema(R) \cap \mathcal{V}_{i-1}]$. 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.

B Proof of Lemma 4.1

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

Case $\boldsymbol{\lambda > \sqrt{m}}$. For every edge $(X, Y) \in G_{\mathsf{DC}} = (V_P, E_P)$, define two variables: $x_{X,Y}$ and $z_{X,Y}$. Jayaraman et al. [19] showed that, for $\lambda > \sqrt{m}$, $polymat(\mathsf{DC})$ is, up to a constant factor, the optimal value of the following LP (named LP$^{(+)}$ following [19]):

LP$^{(+)}$ [19] $\min \sum_{(X,Y)\in E_P} x_{X,Y} \log m + z_{X,Y} \log \lambda$ subject to

$$\sum_{(X,A)\in E_P} (x_{X,A} + z_{X,A}) + \sum_{(A,Y)\in E_P} x_{A,Y} \geq 1 \qquad \forall A \in V_P$$
$$x_{X,Y} \geq 0,\ z_{X,Y} \geq 0 \qquad \forall (X, Y) \in E_P$$

Lemma B.1. *There exists an optimal solution to LP$^{(+)}$ satisfying the condition that the edges in $\{(X, Y) \in E_P \mid z_{X,Y} > 0\}$ induce an acyclic subgraph of G_{DC}.*

We note that while the above lemma is not expressly stated in [19], it can be extrapolated from the analysis presented in Section H.2 of [19]. Nevertheless, the argument laid out in [19] is quite intricate. Our proof, which will be presented below, incorporates news ideas beyond their argument and is considerably shorter. Specifically, these new ideas are evidenced in the way we formulate a novel LP optimal solution in (23)-(26).

Proof of Lemma B.1. Consider an arbitrary optimal solution to $\text{LP}^{(+)}$ that sets $x_{X,Y} = x^*_{X,Y}$ and $z_{X,Y} = z^*_{X,Y}$ for each $(X,Y) \in E_P$. If the edge set $\{(X,Y) \in E_P \mid z^*_{X,Y} > 0\}$ induces an acyclic graph, we are done. Next, we consider that G_{DC} contains a cycle.

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

$$\begin{aligned} x'_{A_2,A_3} &= x^*_{A_2,A_3} + z^*_{A_1,A_2} && (23)\\ x'_{A_1,A_2} &= x^*_{A_1,A_2} && (24)\\ z'_{A_2,A_3} &= 0 && (25)\\ z'_{A_1,A_2} &= 0 && (26) \end{aligned}$$

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

$$\begin{aligned} &\Big(\sum_{(X,Y)\in E_P} x'_{X,Y} \log m + z'_{X,Y} \log \lambda\Big) - \Big(\sum_{(X,Y)\in E_P} x^*_{X,Y} \log m + z^*_{X,Y} \log \lambda\Big) \\ =\ & z^*_{A_1,A_2} \log m - (z^*_{A_1,A_2} + z^*_{A_2,A_3}) \log \lambda \\ \leq\ & z^*_{A_1,A_2} \log m - 2 \cdot z^*_{A_1,A_2} \log \lambda \\ <\ & 0 \qquad (27) \end{aligned}$$

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

We now build a set DC' of degree constraints as follows. First, take an optimal solution $\{x^*_{X,Y}, z^*_{X,Y} \mid (X,Y) \in E_P\}$ to $\text{LP}^{(+)}$ promised by Lemma B.1. Add to DC' a constraint $(X, \{X,Y\}, \lambda)$ for every $(X,Y) \in E_P$ satisfying $z^*_{X,Y} > 0$. Then, for every edge $(X,Y) \in E_P$, add to DC' a constraint $(\emptyset, \{X,Y\}, m)$. The DC' thus constructed must be acyclic. Denote by $G_{\mathsf{DC}'} = (V'_P, E'_P)$ the degree constraint graph of DC'. Note that $V_P = V'_P$ and $E'_P \subset E_P$.

Lemma B.2. *The* DC' *constructed in the above manner satisfies* $\textit{polymat}(\mathsf{DC}') = \Theta(\textit{polymat}(\mathsf{DC}))$.

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

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

$$\textbf{LP}_{\mathbf{1}}^{\mathbf{(+)}} \qquad \min \sum_{(X,Y)\in E_P} x_{X,Y}\log m + z_{X,Y}\log\lambda \text{ subject to}$$

$$\begin{aligned}
&\sum_{(X,A)\in E_P} x_{X,A} + \sum_{(A,Y)\in E_P} x_{A,Y} + \sum_{(X,A)\in E'_P} z_{X,A} \geq 1 && \forall A \in \mathcal{V}_P\\
&x_{X,Y} \geq 0,\ z_{X,Y} \geq 0 \qquad \forall (X,Y) \in E_P
\end{aligned}$$

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

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

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

Consider another LP:

$$\textbf{LP}_{\mathbf{2}}^{\mathbf{(+)}} \qquad \min \sum_{(X,Y)\in E_P} x_{X,Y}\log m + \sum_{(X,Y)\in E'_P} z_{X,Y}\log\lambda \text{ subject to}$$

$$\begin{aligned}
&\sum_{(X,A)\in E_P} x_{X,A} + \sum_{(A,Y)\in E_P} x_{A,Y} + \sum_{(X,A)\in E'_P} z_{X,A} \geq 1 && \forall A \in \mathcal{V}_P\\
&x_{X,Y} \geq 0 \qquad \forall (X,Y) \in E_P\\
&z_{X,Y} \geq 0 \qquad \forall (X,Y) \in E'_P
\end{aligned}$$

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

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

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

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

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

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

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

Next, we review an expression about *polymat*(DC) derived in [19]. Find all the strongly connected components (SCCs) of $G_{\mathsf{DC}} = (V_P, E_P)$. Adopting terms from [19], an SCC is classified as (i) a *source* if it has no in-coming edge from another SCC, or a *non-source* otherwise; (ii) *trivial* if it consists of a single vertex, or *non-trivial* otherwise. Define:

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

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

- S_1 = the set of vertices in S each serving as the center of some directed star in the cover.
- $S_2 = S \setminus S_1$.
- T_2 = the set of vertices in T each serving as the center of some directed star in the cover.
- $T_1 = T \setminus T_2$.

Note that the meanings of the symbols S_1, S_2, T_1, and T_2 follow exactly those in [19] for the reader's convenience (in particular, note the semantics of T_1 and T_2).

We now introduce three quantities:

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

Jayaraman et al. [19] showed:

$$polymat_{dir}(m, \lambda, P) \quad = \quad \Theta\Big(m^{c_1+|S|} \cdot \lambda^{n_1+n_2+|T_1|-2c_1-|S_1|}\Big). \tag{28}$$

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

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

- In each non-trivial non-source SCC, every vertex has an in-coming edge included in E'_P.

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

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

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

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

We will now proceed to argue that $\mathit{polymat}(\mathsf{DC}') = O(\mathit{polymat}(\mathsf{DC}))$. Recall that $\log(\mathit{polymat}(\mathsf{DC}'))$ is the optimal value of the dual modular LP of DC' (see Section 2). On the other hand, the value of $\mathit{polymat}(\mathsf{DC})$ satisfies (28). In the following, we will construct a feasible solution to the dual modular LP of DC' under which the LP's objective function achieves the value of

$$\Big((c_1 + |S|) \cdot \log m\Big) + (n_1 + n_2 + |T_1| - 2c_1 - |S_1|) \cdot \log \lambda \tag{29}$$

which will be sufficient for proving Lemma B.1.

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

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

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

It is tedious but straightforward to verify that all the constraints of the dual modular LP are fulfilled. To confirm that the objective function indeed evaluates to (29), observe:

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

We now conclude the whole proof of Lemma 4.1.

C Proof of Lemma 5.1

By definition, $\log(\mathit{polymat}_{undir}(m, \lambda, P))$ equals the optimal value of the following LP:

Polymatroid LP: maximize $h(V_P)$, from all set functions $h(\cdot)$ over V_P, subject to
(I) $h(\emptyset) = 0$
(II) $h(\{X,Y\}) \le \log m \quad \forall \{X,Y\} \in E_P$
(III) $h(\{X,Y\}) - h(\{X\}) \le \log \lambda \quad \forall \{X,Y\} \in E_P$
(IV) $h(\mathcal{X} \cup \mathcal{Y}) + h(\mathcal{X} \cap \mathcal{Y}) \le h(\mathcal{X}) + h(\mathcal{Y}) \quad \forall \mathcal{X}, \mathcal{Y} \subseteq V_P$
(V) $h(\mathcal{X}) \le h(\mathcal{Y}) \quad \forall \mathcal{X} \subseteq \mathcal{Y} \subseteq V_P$

To see that the above is an instance of linear programming, one can view a set function h over V_P as a point in a $2^{|V_P|}$-dimensional space, where each dimension is a different subset of V_P. Consequently, for any subset $\mathcal{X} \subseteq V_P$, $h(\mathcal{X})$ can be regarded as the "coordinate" of this point on the dimension $\mathcal{X}$.

C.1 Upper Bounding the Objective Function

First, we will show that (22) is an upper bound of $\mathit{polymat}_{undir}(m, \lambda, P)$. More precisely, for every feasible solution $h(.)$ to the polymatroid LP, we will prove:

$$h(V_P) \le \log m + (k-2)\log \lambda \tag{30}$$
$$h(V_P) \le (k_{cycle}/2 + \beta)\log m + (k_{star} - 2\beta)\log \lambda. \tag{31}$$

This implies that $\exp_2(h(V_P))$ is always bounded by the right hand side of (22) and, hence, so is $\mathit{polymat}_{undir}(m, \lambda, P)$.

Proof of (30). Let us order the vertices in V_P as $A_1, A_2, ..., A_k$ with the property that, for each $i \in [2,k]$, the vertex A_i is adjacent in P to at least one vertex A_j with $j < i$. We will denote this value of j as $\mathit{back}(i)$. Such an ordering definitely exists because P is connected. For $i \ge 3$, define $A_{[i]} = \{A_1, A_2, ..., A_i\}$.

To start with, let us observe that, by the constraint (IV) of the polymatroid LP, the inequality

$$h(A_{[i]}) + h(A_{\mathit{back}(i)}) \le h(A_{[i-1]}) + h(A_{\mathit{back}(i)}, A_i) \tag{32}$$

holds for all $i \in [2,k]$. Thus:

$$h(V_P) = h(A_1, A_2) + \sum_{i=3}^{k} h(A_{[i]}) - h(A_{[i-1]})$$

$$\begin{aligned}
\text{(by (32))} \quad &\le \quad h(A_1, A_2) + \sum_{i=3}^{k} h(A_{back(i)}, A_i) - h(A_{back(i)}) \\
&\le \quad \log m + (k-2)\log\lambda.
\end{aligned}$$

The last step used constraints (II) and (III) of the polymatroid LP.

Proof of (31)**.** For each cycle $\pentagon_i$ ($i \in [\alpha]$) in the fractional edge-cover decomposition of P, define $V(\pentagon_i)$ as the set of vertices in $\pentagon_i$ and set $k(\pentagon_i) = |V(\pentagon_i)|$. Likewise, for each star $\star_j$ ($j \in [\beta]$), define $V(\star_j)$ as the set of vertices in $\star_j$ and set $k(\star_j) = |V(\star_j)|$. We aim to prove

$$\text{for each } i \in [\alpha]\text{: } h(V(\pentagon_i)) \le (k(\pentagon_i)/2)\log m; \tag{33}$$

$$\text{for each } j \in [\beta]\text{: } h(V(\star_j)) \le \log m + (k(\star_j) - 2)\log\lambda. \tag{34}$$

Once this is done, we can establish (31) as follows. First, from constraint (IV), we know $h(\mathcal{X} \cup \mathcal{Y}) \le h(\mathcal{X}) + h(\mathcal{Y})$ for any disjoint $\mathcal{X}, \mathcal{Y} \subseteq V_P$; call this *the disjointness rule*. Then, we can derive

$$\begin{aligned}
h(V_P) \quad &\le \quad \sum_{i=1}^{\alpha} h(V(\pentagon_i)) + \sum_{j=1}^{\beta} h(V(\star_j)) \text{ (disjointness rule)} \\
\text{(by (33), (34))} \quad &\le \quad \sum_{i=1}^{\alpha} \frac{k(\pentagon_i)}{2}\log m + \sum_{j=1}^{\beta}(\log m + (k(\star_j) - 2)\log\lambda) \\
&= \quad (k_{cycle}/2 + \beta)\log m + (k_{star} - 2\beta)\log\lambda
\end{aligned}$$

as desired.

We proceed to prove (33). Let us arrange the vertices of $\pentagon_i$ in clockwise order as $X_1, X_2, ..., X_{k(\pentagon_i)}$. For $t \ge 3$, define $X_{[t]} = \{X_1, X_2, ..., X_t\}$. Applying the disjointness rule, we get

$$h(V(\pentagon_i)) \quad \le \quad h(X_1, X_{k(\pentagon_i)}) + \sum_{t=2}^{k(\pentagon_i)-1} h(X_t). \tag{35}$$

Equipped with the above, we can derive

$$\begin{aligned}
& k(\pentagon_i) \cdot \log m \text{ (by constraint (II))} \\
& \ge \Big(\sum_{t=1}^{k(\pentagon_i)-1} h(X_t, X_{t+1}) \Big) + h(\{X_{k(\pentagon_i)}, X_1\}) \\
& \text{(the next few steps will apply constraint (IV))} \\
& \ge h(X_{[3]}) + h(X_2) + \Big(\sum_{t=3}^{k(\pentagon_i)-1} h(X_t, X_{t+1}) \Big) + h(X_{k(\pentagon_i)}, X_1)
\end{aligned}$$

$$\ge h(X_{[4]}) + h(X_2) + h(X_3) + \Big(\sum_{t=4}^{k(\pentagon_i)-1} h(X_t, X_{t+1}) \Big) + h(X_{k(\pentagon_i)}, X_1)$$

...

$$\geq h(X_{[k(\pentagon_i)]}) + \Big(\sum_{t=2}^{k(\pentagon_i)-1} h(X_t) \Big) + h(X_{k(\pentagon_i)}, X_1)$$

(the next step applies (35) and $V(\pentagon_i) = X_{[k(\pentagon_i)]}$)

$$\geq 2 \cdot h(V(\pentagon_i))$$

as claimed in (33).

It remains to prove (34). Let us label the vertices in $\star_j$ as $Y_1, Y_2, ..., Y_{k(\star_j)}$ with Y_1 being the center vertex. For $t \geq 3$, define $Y_{[t]} = \{Y_1, Y_2, ..., Y_t\}$. Then:

$$\begin{aligned} h(V(\star_j)) &= h(Y_1, Y_2) + \sum_{t=3}^{k(\star_j)} h(Y_{[t]}) - h(Y_{[t-1]}) \\ \text{(by constraint (IV))} &\leq h(Y_1, Y_2) + \sum_{t=3}^{k(\star_j)} (h(Y_1, Y_t) - h(Y_1)) \\ \text{(by constraints (II), (III))} &\leq \log m + (k(\star_j) - 2) \log \lambda \end{aligned}$$

as claimed in (34).

C.2 Constructing an Optimal Set Function

To prove Lemma 5.1, we still need to show that $\mathit{polymat}_{undir}(m, \lambda, P)$ is at least the right hand side of (22). For this purpose, it suffices to prove (i) when $\lambda \leq \sqrt{m}$, there is a feasible set function h^* whose $h^*(V_P)$ equals the right hand side of (30), and (ii) when $\lambda > \sqrt{m}$, there is a feasible set function h^* whose $h^*(V_P)$ equals the right hand side of (31). This subsection will construct these set functions explicitly.

Case $\lambda \leq \sqrt{m}$. In this scenario, the function h^* is easy to design. For each $\mathcal{X} \subseteq V_P$, set

$$h^*(\mathcal{X}) = \begin{cases} 0 & \text{if } \mathcal{X} = \emptyset \\ \log m + (|\mathcal{X}| - 2) \log \lambda & \text{otherwise} \end{cases} \tag{36}$$

Obviously, $h^*(V_P) = \log m + (|V_P| - 2) \log \lambda$, as needed. It remains to explain why this h^* is a feasible solution to the polymatroid LP. We prove as follows.

Constraints (I), (II), and (III) are trivial to verify and omitted. Regarding (IV), first note that the constraint is obviously true if $\mathcal{X}$ or $\mathcal{Y}$ is empty. If neither of them is empty, we can derive:

$$\begin{aligned} h^*(\mathcal{X} \cup \mathcal{Y}) + h^*(\mathcal{X} \cap \mathcal{Y}) &= 2 \log m + (|\mathcal{X} \cup \mathcal{Y}| + |\mathcal{X} \cap \mathcal{Y}| - 4) \log \lambda \\ &= 2 \log m + (|\mathcal{X}| + |\mathcal{Y}| - 4) \log \lambda \\ &= h^*(\mathcal{X}) + h^*(\mathcal{Y}) \end{aligned} \tag{37}$$

as needed. Now, consider constraint (V). If $\mathcal{X} = \emptyset$, the constraint holds because $h^*(\mathcal{Y}) = \log m + (|\mathcal{Y}| - 2) \log \lambda \geq \log m - \log \lambda \geq 0$. If $X \neq \emptyset$, then

$$\begin{aligned} h^*(\mathcal{X}) &= \log m + (|\mathcal{X}| - 2) \log \lambda \\ &\leq \log m + (|\mathcal{Y}| - 2) \log \lambda \\ &= h^*(\mathcal{Y}). \end{aligned}$$

Case $\lambda > \sqrt{m}$. Let us look at the fractional edge-cover decomposition of P: $(\pentagon_1, ..., \pentagon_\alpha, \star_1, ..., \star_\beta)$. As before, for each $i \in [\alpha]$, define $V(\pentagon_i)$ as the set of vertices in the cycle $\pentagon_i$; for each $j \in [\beta]$, define $V(\star_j)$ as the set of vertices in the star $\star_j$.

To design the function h^*, for each $\mathcal{X} \subseteq V_P$, we choose the value $h^*(\mathcal{X})$ by the following rules.

- (Rule 1): If $\mathcal{X} = \emptyset$, then $h^*(\mathcal{X}) = 0$.
- (Rule 2): if $\mathcal{X} \subseteq V(\pentagon_i)$ for some $i \in [\alpha]$ but $\mathcal{X} \neq \emptyset$, then $h^*(\mathcal{X}) = \frac{|\mathcal{X}|}{2} \log m$.
- (Rule 3): if $\mathcal{X} \subseteq V(\star_j)$ for some $j \in [\beta]$ but $\mathcal{X} \neq \emptyset$, then $h^*(\mathcal{X}) = \log m + (|\mathcal{X}| - 2) \log \lambda$.
- (Rule 4): Suppose that none of the above rules applies. For each $i \in [\alpha]$, define $\mathcal{Y}_i = \mathcal{X} \cap V(\pentagon_i)$; similarly, for each $j \in [\beta]$, define $\mathcal{Z}_j = \mathcal{X} \cap V(\star_j)$. Then:

$$h^*(\mathcal{X}) \quad = \quad \sum_{i=1}^{\alpha} h^*(\mathcal{Y}_i) + \sum_{j=1}^{\beta} h^*(\mathcal{Z}_j). \tag{38}$$

  The above equation is well-defined because each $h^*(\mathcal{Y}_i)$ can be computed using Rules 1 and 2, and each $h^*(\mathcal{Z}_j)$ can be computed using Rules 1 and 3.

As a remark, our construction ensures that (38) holds for all $\mathcal{X} \subseteq V_P$.

It is easy to check that $h^*(V_P) = \sum_{i=1}^{\alpha} h^*(V(\pentagon_i)) + \sum_{j=1}^{\beta} h^*(V(\star_j))$, which is $(k_{cycle}/2 + \beta) \log m + (k_{star} - 2\beta) \log \lambda$, as needed. It suffices to verify the feasibility of h^*.

Constraint (I) is guaranteed by Rule 1. Next, we will discuss constraints (II) and (III) together. The verification is easy (and hence omitted) if $\{X, Y\}$ is a cycle edge or a star edge. Now, consider that $\{X, Y\}$ is neither a cycle edge nor a star edge. By the properties of fractional edge-cover decomposition, one of the following must occur:

- (C-1) X and Y are in two different cycles;
- (C-2) X and Y are in two different stars;
- (C-3) one of X and Y is in a cycle and the other is in a star.

In all the above scenarios, it holds that $h^*(\{X, Y\}) = h^*(\{X\}) + h^*(\{Y\})$. Thus, to confirm (II), it suffices to show $h^*(\{X\}) + h^*(\{Y\}) \leq \log m$, and to confirm (III), it suffices to show $h(\{Y\}) \leq \log \lambda$. It is rudimentary to verify both of these inequalities using Rules 2 and 3 and the fact $\log m < 2 \log \lambda$.

Constraint (IV) is trivially true if either $\mathcal{X}$ or $\mathcal{Y}$ is empty. Now, assume that neither is empty. If $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{V}(\pentagon_i)$ for some $i \in [\alpha]$, we have:

$$\begin{aligned} h^*(\mathcal{X} \cup \mathcal{Y}) + h^*(\mathcal{X} \cap \mathcal{Y}) &= \frac{|\mathcal{X} \cap \mathcal{Y}| + |\mathcal{X} \cup \mathcal{Y}|}{2} \log m \\ &= \frac{|\mathcal{X}| + |\mathcal{Y}|}{2} \log m \\ &= h^*(\mathcal{X}) + h^*(\mathcal{Y}). \end{aligned}$$

If $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{V}(\star_j)$ for some $j \in [\beta]$, the reader can verify (IV) with the same derivation in (37).

Next, we consider arbitrary $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{V}$. Define for each $i \in [\alpha]$ and $j \in [\beta]$:

$$\mathcal{X}_i^{cycle} \quad = \quad \mathcal{X} \cap \mathcal{V}(\pentagon_i) \tag{39}$$

$$\begin{aligned}\mathcal{X}_j^{star} &= \mathcal{X} \cap \mathcal{V}(\star_j) &(40)\\ \mathcal{Y}_i^{cycle} &= \mathcal{Y} \cap \mathcal{V}(\pentagon_i) &(41)\\ \mathcal{Y}_j^{star} &= \mathcal{Y} \cap \mathcal{V}(\star_j). &(42)\end{aligned}$$

We can derive:

$$\begin{aligned} & h^*(\mathcal{X} \cup \mathcal{Y}) \\ \text{(by Rule 4)} \quad = & \sum_{i=1}^{\alpha} h^*((\mathcal{X} \cup \mathcal{Y}) \cap \mathcal{V}(\pentagon_i)) + \sum_{j=1}^{\beta} h^*((\mathcal{X} \cup \mathcal{Y}) \cap \mathcal{V}(\star_j)) \\ = & \sum_{i=1}^{\alpha} h^*(\mathcal{X}_i^{cycle} \cup \mathcal{Y}_i^{cycle}) + \sum_{j=1}^{\beta} h^*(\mathcal{X}_j^{star} \cup \mathcal{Y}_j^{star}) \end{aligned}$$

and similarly:

$$\begin{aligned} & h^*(\mathcal{X} \cap \mathcal{Y}) \\ = & \sum_{i=1}^{\alpha} h^*((\mathcal{X} \cap \mathcal{Y}) \cap \mathcal{V}(\pentagon_i)) + \sum_{j=1}^{\beta} h^*((\mathcal{X} \cap \mathcal{Y}) \cap \mathcal{V}(\star_j)) \\ = & \sum_{i=1}^{\alpha} h^*(\mathcal{X}_i^{cycle} \cap \mathcal{Y}_i^{cycle}) + \sum_{j=1}^{\beta} h^*(\mathcal{X}_j^{star} \cap \mathcal{Y}_j^{star}). \end{aligned}$$

Recall that (IV) has been validated in the scenario where (i) $\mathcal{X}$ and $\mathcal{Y}$ are contained in the same cycle, or (ii) they are contained in the same star. Thus, it holds for each $i \in [\alpha]$ and $j \in [\beta]$ that

$$\begin{aligned} h^*(\mathcal{X}_i^{cycle} \cup \mathcal{Y}_i^{cycle}) + h^*(\mathcal{X}_i^{cycle} \cap \mathcal{Y}_i^{cycle}) &\le h^*(\mathcal{X}_i^{cycle}) + h^*(\mathcal{Y}_i^{cycle}) &(43)\\ h^*(\mathcal{X}_j^{star} \cup \mathcal{Y}_j^{star}) + h^*(\mathcal{X}_j^{star} \cap \mathcal{Y}_j^{star}) &\le h^*(\mathcal{X}_j^{star}) + h^*(\mathcal{Y}_j^{star}). &(44)\end{aligned}$$

Equipped with these facts, we get:

$$\begin{aligned} & h^*(\mathcal{X} \cup \mathcal{Y}) + h^*(\mathcal{X} \cap \mathcal{Y}) \\ = & \sum_{i=1}^{\alpha} h^*(\mathcal{X}_i^{cycle} \cup \mathcal{Y}_i^{cycle}) + h^*(\mathcal{X}_i^{cycle} \cap \mathcal{Y}_i^{cycle}) + \\ & \sum_{j=1}^{\beta} h^*(\mathcal{X}_j^{star} \cup \mathcal{Y}_j^{star}) + h^*(\mathcal{X}_j^{star} \cap \mathcal{Y}_j^{star}) \\ \text{(by (43) and (44))} \quad \le & \sum_{i=1}^{\alpha} h^*(\mathcal{X}_i^{cycle}) + h^*(\mathcal{Y}_i^{cycle}) + \sum_{j=1}^{\beta} h^*(\mathcal{X}_j^{star}) + h^*(\mathcal{Y}_j^{star}) \\ \text{(by Rule 4)} \quad = & h^*(\mathcal{X}) + h^*(\mathcal{Y}). \end{aligned}$$

This verifies the correctness of constraint (IV).

Finally, let us look at (V). This constraint is trivially met if $\mathcal{X} = \emptyset$. Next, we assume that $\mathcal{X}$ is not empty. If $\mathcal{Y}$ is a subset of $\mathcal{V}(\pentagon_i)$ for some $i \in [\alpha]$, then, by Rule 2, $h^*(\mathcal{X}) = \frac{|\mathcal{X}|}{2} \log m \le \frac{|\mathcal{Y}|}{2} \log m = h^*(\mathcal{Y})$. If $\mathcal{Y}$ is a subset of $\mathcal{V}(\star_j)$ for some $j \in [\beta]$, then, by Rule 3, $h^*(\mathcal{X}) = \log m + (|\mathcal{X}| - 2) \log \lambda \le \log m + (|\mathcal{Y}| - 2) \log \lambda = h^*(\mathcal{Y})$.

It remains to consider the situation where $\mathcal{Y}$ can be any subset of $\mathcal{V}$. Using the definitions in (39)-(42), we have:

$$\begin{aligned}
h^*(\mathcal{X}) &= \sum_{i=1}^{\alpha} h^*(\mathcal{X}_i^{cycle}) + \sum_{j=1}^{\beta} h^*(\mathcal{X}_j^{star}) \quad \text{(by Rule 4)} \\
&\leq \sum_{i=1}^{\alpha} h^*(\mathcal{Y}_i^{cycle}) + \sum_{j=1}^{\beta} h^*(\mathcal{Y}_j^{star}) \\
&\quad \text{(we have verified (V) in the scenario where } \mathcal{Y} \text{ is contained in a cycle or a star)} \\
\text{(by Rule 4)} &= h^*(\mathcal{Y})
\end{aligned}$$

as needed to verify constraint (V).

D Equation (22) as an Output Size Bound and Its Tightness

We will start by proving that (22) is an asymptotic upper bound on $|occ(G, P)|$ for any graph G that has m edges and a maximum vertex-degree at most λ. First, we will prove that $|occ(G, P)| = O(m \cdot \lambda^{k-2})$ (recall that k is the number of vertices in P). Identify an arbitrary spanning tree $\mathcal{T}$ of the pattern P. It is rudimentary to verify that $\mathcal{T}$ has $O(m \cdot \lambda^{k-2})$ occurrences in G,[5] implying that the number of occurrences of P is also $O(m \cdot \lambda^{k-2})$. Next, we will demonstrate that $|occ(G, P)| = O(m^{\frac{k_{cycle}}{2}+\beta} \cdot \lambda^{k_{star}-2\beta})$. For each $i \in [\alpha]$, let $k(\pentagon_i)$ be the number of vertices in $\pentagon_i$; for each $j \in [\beta]$, let $k(\star_j)$ be the number of vertices in $\star_j$. The fractional edge cover number of $\pentagon_i$ ($i \in [\alpha]$) is $\rho^*(\pentagon_i) = k(\pentagon_i)/2$; this means $\sum_{i=1}^{k} \rho^*(\pentagon_i) = k(\pentagon_i)/2 = k_{cycle}/2$. For each $i \in [\alpha]$, the pattern $\pentagon_i$ can have $O(m^{\rho^*(\pentagon_i)})$ occurrences in G. For each $j \in [\beta]$, by our earlier analysis, the pattern $\star_j$ can have $O(m^{k(\star_j)-2})$ occurrences in G. Thus, the number of occurrences of P must be asymptotically bounded by

$$\prod_{i=1}^{\alpha} m^{\rho^*(\pentagon_i)} \cdot \prod_{j=1}^{\beta} m \cdot \lambda^{k(\star_j)-2} = m^{\frac{k_{cycle}}{2}+\beta} \cdot \lambda^{k_{star}-2\beta}.$$

The rest of this section will concentrate on the tightness of (22) as an upper bound of $|occ(G, P)|$. Our objective is to prove:

Theorem D.1. *Fix an arbitrary integer $k \geq 2$. For any values of m and λ satisfying $m \geq \max\{16k^2, 64\}$ and $k \leq \lambda \leq m/(4k)$, there is an undirected simple graph G^* satisfying all the conditions below:*

- *G^* has at most m edges and and a maximum vertex degree at most λ;*
- *For any undirected simple pattern graph $P = (V_P, E_P)$ that has k vertices, the number of occurrences of P in G^* is at least $\frac{1}{(4k)^k} \cdot polymat_{undir}(m, \lambda, P)$, where $polymat_{undir}(m, \lambda, P)$ is given in (22).*

It is worth noting that we aim to construct a *single* G^* that is a "bad" input for *all* possible patterns (with k vertices) simultaneously. Our proof will deal with the case $k \leq \lambda < \sqrt{m}$ and the case $\sqrt{m} \leq \lambda \leq m/(4k)$ separately.

[5]There are $O(m)$ choices to map an arbitrary edge $\mathcal{T}$ to an edge in G, and then λ choices to map each of the $k-2$ remaining vertices of $\mathcal{T}$ to a vertex in G.

Remark. The main challenge in the proof is to establish the factor $\frac{1}{(4k)^k}$. There exist lower bound arguments [19, 36] that can be used in our context to build a hard instance for each *individual* pattern P. A naive way to form a *single* hard input G^* is to combine the hard instances of all possible patterns having k vertices. But this would introduce a gigantic factor roughly $1/2^{\Omega(k^2)}$.

D.1 Case $k \le \lambda < \sqrt{m}$

In this scenario, G^* only needs to be the graph that consists of $\lfloor m/\binom{\lambda}{2} \rfloor$ independent λ-cliques.[6] An occurrence of P can be formed by mapping V_P to k arbitrary distinct vertices in one λ-clique. The number of occurrences of P in G^* is at least

$$\begin{aligned}
& \lfloor \frac{m}{\binom{\lambda}{2}} \rfloor \cdot \binom{\lambda}{k} \\
\text{(applying } \binom{\lambda}{k} \ge (\lambda/k)^k) \quad \ge \quad & \Big(\frac{2m}{\lambda^2 - \lambda} - 1\Big) \cdot (\lambda/k)^k \\
\text{(as } m > \lambda^2 - \lambda) \quad > \quad & \frac{m}{\lambda^2 - \lambda} \cdot (\lambda/k)^k \\
\text{(as } k \ge 2) \quad > \quad & \frac{1}{k^k} \cdot m \cdot \lambda^{k-2} = \frac{1}{k^k} \cdot \mathit{polymat}_{\mathit{undir}}(m, \lambda, P).
\end{aligned}$$

D.2 Case $\sqrt{m} \le \lambda \le m/(4k)$

We construct G^* as follows. First, create three disjoint sets of vertices: V_A, V_B, and V_C, whose sizes are $\lceil \lambda/4 \rceil$, $\lceil m/(4\lambda) \rceil$, and $\lceil \sqrt{m}/4 \rceil$. Because $\lambda \le \frac{m}{4k}$ and $m \ge 16k^2$, each of V_A, V_B, and V_C has a size at least k. Then, decide the edges of G^* in the way below:

- Add an edge between each pair of vertices in $V_B \cup V_C$ (thereby producing a clique of $\lceil m/(4\lambda) \rceil + \lceil \sqrt{m}/4 \rceil$ vertices).
- Add an edge $\{u, v\}$ for each vertex pair $(u, v) \in V_A \times V_B$.

Next, we show that G^* has at most m edges through a careful calculation. First, the number of edges between V_A and V_B is

$$\begin{aligned}
& \lceil \lambda/4 \rceil \cdot \lceil m/(4\lambda) \rceil \\
\le \quad & (\lambda/4 + 1) \cdot (m/(4\lambda) + 1) \\
= \quad & m/16 + \lambda/4 + m/(4\lambda) + 1 \\
\text{(as } \sqrt{m} \le \lambda \le m) \quad \le \quad & 5m/16 + \sqrt{m}/4 + 1 \\
\text{(as } 16 \le m) \quad \le \quad & 7m/16.
\end{aligned}$$

The number of edges between V_B and V_C is:

$$\begin{aligned}
& \lceil m/(4\lambda) \rceil \cdot \lceil \sqrt{m}/4 \rceil \\
\le \quad & (m/(4\lambda) + 1) \cdot (\sqrt{m}/4 + 1) \\
= \quad & m^{1.5}/(16\lambda) + m/(4\lambda) + \sqrt{m}/4 + 1 \\
\text{(as } \sqrt{m} \le \lambda) \quad \le \quad & m/16 + \sqrt{m}/2 + 1 \\
\text{(as } 16 \le m) \quad \le \quad & m/4.
\end{aligned}$$

[6]When $\lambda < \sqrt{m}$, $\binom{\lambda}{2} = \frac{\lambda^2 - \lambda}{2} < m$. Hence, G^* contains at least one clique.

The number of edges between vertices in V_B is:

$$\begin{aligned} & \lceil m/(4\lambda)\rceil \cdot (\lceil m/(4\lambda)\rceil - 1)/2 \\ \leq\ & (m/(4\lambda)+1)\cdot(m/(4\lambda))/2 \\ =\ & m^2/(32\lambda^2) + m/(8\lambda) \\ (\text{as } \sqrt{m} \leq \lambda)\quad \leq\ & m/32 + \sqrt{m}/8 \leq 5m/32 \end{aligned}$$

The number of edges between vertices in V_C is:

$$\begin{aligned} & \lceil \sqrt{m}/4\rceil \cdot (\lceil \sqrt{m}/4 - 1)/2 \\ \leq\ & (\sqrt{m}/4+1)\cdot(\sqrt{m}/4)/2 \\ =\ & m/32 + \sqrt{m}/8 \leq 5m/32. \end{aligned}$$

Thus, the total number of edges in G^* is at most $7m/16 + m/4 + 5m/32 + 5m/32 = m$.

The maximum vertex degree in G^* is decided by the vertices in V_B and equals

$$\begin{aligned} & \lceil \lambda/4\rceil + \lceil m/(4\lambda)\rceil - 1 + \lceil \sqrt{m}/4\rceil \\ \leq\ & \lambda/4 + m/(4\lambda) + \sqrt{m}/4 + 2 \\ (\text{as } \lambda \geq \sqrt{m})\quad \leq\ & 3\lambda/4 + 2 \leq \lambda. \end{aligned}$$

Therefore, G^* satisfies the requirement in the first bullet of Theorem D.1.

The rest of the proof will focus on the theorem's second bullet. Let P be an arbitrary pattern with k vertices, and take a fractional edge-cover decomposition of P: $(\pentagon_1, ..., \pentagon_\alpha, \star_1, ..., \star_\beta)$. Define k_{star} and k_{cycle} in the same way as in (21) and (20). We claim:

Lemma D.2. *There is an integer $s \in [0, \beta]$ under which we can divide the vertices of P into disjoint sets U_A, U_B and U_C such that*

- $|U_A| = k_{star} - \beta - s$, $|U_B| = \beta - s$, $|U_C| = k_{cycle} + 2s$;
- *P has no edges between U_A and U_C;*
- *P has no edges between any two vertices in U_A.*

The proof of the lemma is non-trivial and deferred to the end of this section. An occurrence of P in G^* can be formed through the three steps below:

1. Map U_A to $|U_A|$ distinct vertices in V_A.
2. Map U_B to $|U_B|$ distinct vertices in V_B.
3. Map U_C to $|U_C|$ distinct vertices in V_C.

Each step is carried out independently from the other steps. Hence, the number of occurrences of P in G^* is at least

$$\begin{aligned} & \binom{\lambda/4}{k_{star}-\beta-s} \cdot \binom{\frac{m}{4\lambda}}{\beta - s} \cdot \binom{\sqrt{m}/4}{k_{cycle}+2s} \\ \geq\ & \Big(\frac{\lambda/4}{k_{star}-\beta-s}\Big)^{k_{star}-\beta-s} \cdot \Big(\frac{\frac{m}{4\lambda}}{\beta-s}\Big)^{\beta-s} \cdot \Big(\frac{\sqrt{m}/4}{k_{cycle}+2s}\Big)^{k_{cycle}+2s} \end{aligned}$$

$$
\begin{aligned}
&\geq \frac{\lambda^{k_{star}-\beta-s}}{4^{k_{star}-\beta-s}\cdot k^{k_{star}-\beta-s}} \cdot \frac{m^{\beta-s}}{4^{\beta-s}\cdot\lambda^{\beta-s}\cdot k^{\beta-s}} \cdot \frac{m^{k_{cycle}/2+s}}{4^{k_{cycle}+2s}\cdot k^{k_{cycle}+2s}} \\
&= \frac{1}{(4k)^k}\cdot m^{\frac{k_{cycle}}{2}+\beta}\cdot\lambda^{k_{star}-2\beta} = \frac{1}{(4k)^k}\cdot polymat_{undir}(m,\lambda,P).
\end{aligned}
$$

Proof of Lemma D.2. With each vertex X in the pattern graph $P = (V_P, E_P)$, we associate a variable $\nu(X) \geq 0$ (equivalently, ν is a function from V_P to $\mathbb{R}_{\geq 0}$). Consider the following LP defined on these variables.

vertex-pack LP $\quad \max \sum_{X\in V_P} \nu(X)$ subject to

$$
\begin{aligned}
&\sum_{X:X\in F}\nu(X)\leq 1 && \forall F\in E_P\\
&\nu(X)\geq 0 && \forall X\in V_P
\end{aligned}
$$

A feasible solution ν is said to be *half-integral* if $\nu(X) \in \{0, \frac{1}{2}, 1\}$ for each $X \in V_P$.

Consider any fractional edge-cover decomposition of P: $(\pentagon_1, ..., \pentagon_\alpha, \star_1, ..., \star_\beta)$. Recall that each star $\star_j$ ($j \in [\beta]$) contains a vertex designated as the center. We refer to every other vertex in the star as a *petal*.

Lemma D.3. *For the vertex-pack LP, there exists a half-integral optimal solution $\{\nu^*(X) \mid X \in V_P\}$ satisfying all the conditions below.*

- $\nu^*(X) = 0.5$ *for every vertex X in $\pentagon_1, ...\pentagon_\alpha$.*
- *For every $j \in [\beta]$, the function ν^* has the properties below.*
  - *If $\star_j$ has at least two edges, then $\nu^*(X) = 0$ holds for the star's center X and $\nu^*(Y) = 1$ holds for every petal Y.*
  - *If $\star_j$ has only one edge $\{X, Y\}$ — in which case we call $\star_j$ a "one-edge star" — then $\nu^*(X) + \nu^*(Y) = 1$.*

Proof. Let us first define the dual of vertex-pack LP. Associate each edge $F \in E_P$ with a variable $w(F) \geq 0$ (equivalently, w is a function from E_P to $\mathbb{R}_{\geq 0}$). Then, the dual is:

edge-cover LP $\quad \min \sum_{F\in E_P} w(F)$ subject to

$$
\begin{aligned}
&\sum_{F\in E_P:X\in F} w(F)\geq 1 && \forall X\in V_P\\
&w(F)\geq 0 && \forall F\in E_P
\end{aligned}
$$

The given fractional edge-cover decomposition implies an optimal solution $\{w^*(F) \mid F \in E_p\}$ to the edge-cover LP satisfying:

- $w^*(F) = 0.5$ for every edge F in $\pentagon_1, ...\pentagon_\alpha$;
- $w^*(F) = 1$ for every edge F in $\star_1, ..., \star_\beta$;
- $w^*(F) = 0$ for every other edge $F \in E_p$.

By the complementary slackness theorem, the function w^* implies an optimal solution $\{\nu'(X) \mid X \in V_P\}$ to the vertex-pack LP with the properties below.

- **P1**: For every edge $\{X, Y\} \in E_p$, if $w^*(\{X, Y\}) > 0$, then $\nu'(X) + \nu'(Y) = 1$.

- **P2**: For every vertex X satisfying $\sum_{X \in F} w^*(F) > 1$, $\nu'(X) = 0$.

From the above, we can derive additional properties of the solution $\{\nu'(X) \mid X \in V_P\}$.

- **P3**: $\nu'(X) = 0.5$ for every vertex X in $\pentagon_1, ..., \pentagon_\alpha$. To see why, consider any $\pentagon_i$ ($i \in [\alpha]$). Let $X_1, X_2, ..., X_\ell$ (for some $\ell \geq 3$) be the vertices of $\pentagon_i$ in clockwise order. Property **P1** yields ℓ equations: $\nu'(X_j) + \nu'(X_{j+1}) = 1$ for $j \in [\ell - 1]$ and $\nu'(X_\ell) + \nu'(X_1) = 1$. Solving the system of these equations gives $\nu'(X_j) = 0.5$ for each $j \in [\ell]$.

- For every $j \in [\beta]$, we have:

  - **P4-1**: If $\star_j$ has at least two edges, then $\nu'(X) = 0$ holds for the star's center X and $\nu'(Y) = 1$ for every petal Y. To see why, notice that $\sum_{F \in E_P : X \in F} w^*(F)$ is precisely the number of edges in $\star_j$ and, hence, $\sum_{F \in E_P : X \in F} w^*(F) > 1$. Thus, **P2** asserts $\nu'(X) = 0$. It then follows from **P1** that $\nu'(Y) = 1 - \nu'(X) = 1$ for every petal Y.
  - **P4-2**: If $\star_j$ has only one edge $\{X, Y\}$, then $\nu'(X) + \nu'(Y) = 1$. This directly follows from **P1** and the fact that $w^*(\{X, Y\}) = 1$.

Now, we show how to construct $\nu^*(.)$. If $\nu'(.)$ is already half-integral, then we set $\nu^*(X) = \nu'(X)$ for all $X \in V_P$ and finish. Otherwise, set

- $\nu^*(X) = \nu'(X)$ for every $X \in V_P$ with $\nu'(X) \in \{0, 1, \frac{1}{2}\}$;
- $\nu^*(X) = 0$ for every $X \in V_P$ with $0 < \nu'(X) < 1/2$;
- $\nu^*(X) = 1$ for every $X \in V_P$ with $1/2 < \nu'(X) < 1$.

We need to verify that $\nu^*(.)$ meets the conditions listed in the statement of Lemma D.3. Clearly, $\nu^*(X) = \nu'(X) = 1/2$ for every vertex X in $\pentagon_1, ...\pentagon_\alpha$ (property **P3**). For each $j \in [\beta]$, if $\star_j$ has at least two edges, then $\nu^*(X) = \nu'(X) = 0$ holds for the star's center X and $\nu^*(Y) = \nu'(Y) = 1$ holds for every petal Y (property **P4-1**). If $\star_j$ has only one edge $\{X, Y\}$ — that is, $\star_j$ is a one-edge star — property **P4-2** tells us that $\nu'(X) + \nu'(Y) = 1$. If $\nu'(X) = \nu'(Y) = 1/2$, then $\nu^*(X) + \nu^*(Y) = 1/2 + 1/2 = 1$. Otherwise, one vertex of $\{X, Y\}$ — say X — satisfies $0 < \nu'(X) < 1/2$, and the other vertex satisfies $1/2 < \nu'(Y) < 1$. Our construction ensures that $\nu^*(X) + \nu^*(Y) = 0 + 1 = 1$.

It remains to check that $\{\nu^*(X) \mid X \in V_P\}$ is indeed an optimal solution to the vertex-pack LP. First, we prove the the solution's feasibility. For this purpose, given any edge $F = \{X, Y\} \in E_P$, we must show $\nu^*(X) + \nu^*(Y) \leq 1$. Assume there exists an edge $F = \{X, Y\} \in E_P$ such that $\nu^*(X) + \nu^*(Y) > 1$. Because $\nu^*(.)$ is half-integral, in this situation at least one vertex in $\{X, Y\}$ — say X — must receive value $\nu^*(X) = 1$. We proceed differently depending on the value of $\nu^*(Y)$.

- $\nu^*(Y) = 1/2$. By how ν^* is constructed from ν', in this case we must have $\nu'(X) > 1/2$ and $\nu'(Y) = 1/2$. But then $\nu'(X) + \nu'(Y) > 1$, violating the fact that $\nu'(.)$ is a feasible solution to the vertex-pack LP.

- $\nu^*(Y) = 1$. By how ν^* is constructed, we must have $\nu'(X) > 1/2$ and $\nu'(Y) > 1/2$. Again, $\nu'(X) + \nu'(Y) > 1$, violating the feasibility of $\nu'(.)$.

Finally, we will prove that $\{\nu^*(X) \mid X \in V_P\}$ achieves the optimal value for the vertex-pack LP's objective function. This is true because

$$\sum_{X \in V_P} \nu^*(X) \quad = \quad \Big(\sum_{X : X \text{ not in any one-edge star}} \nu^*(X) \Big) + \sum_{X : X \text{ in a one-edge star}} \nu^*(X)$$

$$\begin{aligned} &= \Big(\sum_{X:X \text{ not in any one-edge star}} \nu'(X) \Big) + \text{number of one-edge stars} \\ \text{(property } \mathbf{P4\text{-}2}\text{)} &= \Big(\sum_{X:X \text{ not in any one-edge star}} \nu'(X) \Big) + \sum_{X:X \text{ in a one-edge star}} \nu'(X) \\ &= \sum_{X \in V_P} \nu'(X) \end{aligned}$$

and $\nu'(.)$ is an optimal solution to the vertex-pack LP. □

We are now ready to prove Lemma D.2. Let $\{\nu^*(X) \mid X \in V_P\}$ be an optimal solution to the vertex-pack LP problem promised by Lemma D.3. Divide V_P into three subsets

$$\begin{aligned} U_A &= \{X \in V_P \mid \nu^*(X) = 1\} \\ U_B &= \{X \in V_P \mid \nu^*(X) = 0\} \\ U_C &= \{X \in V_P \mid \nu^*(X) = 1/2\}. \end{aligned}$$

By the feasibility of $\nu^*(.)$ to the vertex-pack LP, no vertex in U_A can be adjacent to any vertex in U_C, and no two vertices in U_A can be adjacent to each other. It remains to verify that the sizes of U_A, U_B, and U_C meet the requirement in the first bullet of Lemma D.2. To facilitate our subsequent argument, given a one-edge star $\star_j$ (for some $j \in [\beta]$), we call it a *half-half one-edge star* if $\nu^*(X) = \nu^*(Y) = 1/2$, where $\{X, Y\}$ is the (only) edge in $\star_j$. Define

$$s \;=\; \text{number of half-half one-edge stars in } \{\star_1, ..., \star_\beta\}.$$

On the other hand, if a one-edge star $\star_j$ is not half-half, we call it a *0-1 one-edge star*.

By Lemma D.3, U_C includes all the vertices in $\pentagon_1, ..., \pentagon_\alpha$ and all the vertices in the half-half one-edge stars. Hence, $|U_C| = k_{cycle} + 2s$. On the other hand, U_B includes (i) the center of every star that has at least two edges and (ii) exactly one vertex from every 0-1 one-edge star. In other words, every star contributes 1 to the size of U_B except for the half-half one-edge stars, implying $|U_B| = \beta - s$. Finally, $|U_A| = k - |U_B| - |U_C| = k_{star} - \beta - s$. This completes the proof of Lemma D.2.

E Analysis of the Sampling Algorithm in Section 5

The purpose of preprocessing is essentially reorganizing the data graph $G = (V, E)$ in the adjacency-list format, which can be easily done in $O(|E|)$ expected time. The following discussion focuses on the cost of extracting a sample. We will first consider $\lambda \le \sqrt{|E|}$ before attending to $\lambda > \sqrt{|E|}$.

Case $\boldsymbol{\lambda \le \sqrt{|E|}}$. Let $G_{sub} = (V_{sub}, E_{sub})$ be an occurrence of P in G. There exist a constant number c of isomorphism bijections $g : V_P \to V_{sub}$ between P and G_{sub} (the number c is the number of automorphisms of P). Fix any such bijection g. Each repeat of our algorithm builds a map $f : V_p \to V$. It is rudimentary to verify that $\mathbf{Pr}[f = g] = \frac{1}{2|E|} \cdot \frac{1}{\lambda^{k-2}}$. Hence, each occurrence is returned with probability $\frac{c}{2|E|} \cdot \frac{1}{\lambda^{k-2}}$. It is now straightforward to prove that our algorithm has expected sample time: $O(|E| \cdot \lambda^{k-2} / \max\{1, \text{OUT}\})$.

Case $\boldsymbol{\lambda > \sqrt{|E|}}$. If OUT $= 0$, the two-thread approach allows our algorithm to terminate in $O(\mathit{polymat}_{undir}(|E|, \lambda, P)) = O(m^{\frac{k_{cycle}}{2} + \beta} \cdot \lambda^{k_{star} - 2\beta})$ time. Next, we consider only OUT ≥ 1.

For each $i \in [\alpha]$, denote by $k(\pentagon_i)$ the number of vertices in $\pentagon_i$; for each $j \in [\beta]$, denote by $k(\star_j)$ the number of vertices in $\star_j$. The fractional edge cover number of $\pentagon_i$ ($i \in [\alpha]$) is $\rho^*(\pentagon_i) = k(\pentagon_i)/2$. This means $\sum_{i=1}^{k} \rho^*(\pentagon_i) = k(\pentagon_i)/2 = k_{cycle}/2$.

Fix an arbitrary occurrence $G_{sub} = (V_{sub}, E_{sub})$ of P in G. Let g be any of the c isomorphism bijections between P and G_{sub}. Each repeat of our algorithm constructs a map $f : V_P \to V$ through isomorphism bijections $f_{\pentagon_1}, ..., f_{\pentagon_\alpha}, f_{\star_1}, ..., f_{\star_\beta}$. The event $g = f$ happens if and only if all of the following events occur.

- Event $\mathbf{E}_{\pentagon_i}$ ($i \in [\alpha]$): $g(A) = f_{\pentagon_i}(A)$ for each vertex A of $\pentagon_i$;
- Event $\mathbf{E}_{\star_j}$ ($j \in [\beta]$): $g(A) = f_{\star_i}(A)$ for each vertex A of $\star_j$.

If $c_{\pentagon_i}$ ($i \in [\alpha]$) is the number of automorphisms of $\pentagon_i$, we have

$$\mathbf{Pr}[\mathbf{E}_{\pentagon_i}] = \frac{1}{c_{\pentagon_i} \cdot |occ(G, \pentagon_i)|}. \tag{45}$$

Likewise, if $c_{\star_j}$ ($j \in [\beta]$) is the number of automorphisms of $\star_j$, we have

$$\mathbf{Pr}[\mathbf{E}_{\star_j}] = \frac{1}{c_{\star_j} \cdot |occ(G, \star_j)|}. \tag{46}$$

It follows from (45) and (46) that

$$\mathbf{Pr}[g = f] = \Big(\prod_{i=1}^{\alpha} \frac{1}{c_{\pentagon_i} \cdot |occ(G, \pentagon_i)|}\Big) \cdot \Big(\prod_{j=1}^{\beta} \frac{1}{c_{\star_j} \cdot |occ(G, \star_j)|}\Big).$$

Therefore:

$$\mathbf{Pr}[G_{sub} \text{ sampled}] = c \cdot \Big(\prod_{i=1}^{\alpha} \frac{1}{c_{\pentagon_i} \cdot |occ(G, \pentagon_i)|}\Big) \cdot \Big(\prod_{j=1}^{\beta} \frac{1}{c_{\star_j} \cdot |occ(G, \star_j)|}\Big). \tag{47}$$

As this probability is identical for all G_{sub}, we know that each occurrence of P is sampled with the same probability.

The expected number of repeats to obtain a sample occurrence of P is

$$O\Big(\frac{\prod_{i=1}^{\alpha} |occ(G, \pentagon_i)| \cdot \prod_{j=1}^{\beta} |occ(G, \star_j)|}{\text{OUT}}\Big)$$

whereas each repeat runs in time at the order of

$$\Big(\prod_{i=1}^{\alpha} \frac{|E|^{\rho^*(\pentagon_i)}}{|occ(G, \pentagon_i)|}\Big) \cdot \Big(\prod_{j=1}^{\beta} \frac{|E| \cdot \lambda^{k(\star_j)-2}}{|occ(G, \star_j)|}\Big). \tag{48}$$

We can now conclude that the expected sample time is at the order of

$$\frac{1}{\text{OUT}} \cdot \Big(\prod_{i=1}^{\alpha} |E|^{\rho^*(\pentagon_i)}\Big) \cdot \Big(\prod_{j=1}^{\beta} |E| \cdot \lambda^{k(\star_j)-2}\Big) = \frac{|E|^{k_{cycle}/2} \cdot \lambda^{k_{star}-2}}{\text{OUT}}.$$

References

[1] Amir Abboud, Seri Khoury, Oree Leibowitz, and Ron Safier. Listing 4-cycles. *CoRR*, abs/2211.10022, 2022.

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

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

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

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

[6] Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *Innovations in Theoretical Computer Science (ITCS)*, pages 6:1–6:20, 2019.

[7] Albert Atserias, Martin Grohe, and Daniel Marx. Size bounds and query plans for relational joins. *SIAM Journal on Computing*, 42(4):1737–1767, 2013.

[8] Matthias Bentert, Till Fluschnik, Andre Nichterlein, and Rolf Niedermeier. Parameterized aspects of triangle enumeration. *Journal of Computer and System Sciences (JCSS)*, 103:61–77, 2019.

[9] Andreas Bjorklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 223–234, 2014.

[10] Surajit Chaudhuri, Rajeev Motwani, and Vivek R. Narasayya. On random sampling over joins. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 263–274, 1999.

[11] Yu Chen and Ke Yi. Random sampling and size estimation over cyclic joins. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 7:1–7:18, 2020.

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

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

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

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

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

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

[18] Chinh T. Hoang, Marcin Kaminski, Joe Sawada, and R. Sritharan. Finding and listing induced paths and cycles. *Discrete Applied Mathematics*, 161(4-5):633–641, 2013.

[19] Sai Vikneshwar Mani Jayaraman, Corey Ropell, and Atri Rudra. Worst-case optimal binary join algorithms under general ℓ_p constraints. *CoRR*, abs/2112.01003, 2021.

[20] Ce Jin and Yinzhan Xu. Removing additive structure in 3sum-based reductions. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 405–418, 2023.

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

[22] Mahmoud Abo Khamis, Vasileios Nakos, Dan Olteanu, and Dan Suciu. Join size bounds using lp-norms on degree sequences. *CoRR*, abs/2306.14075, 2023.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[38] Todd L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 96–106, 2014.

[39] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. Random sampling over joins revisited. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 1525–1539, 2018.