Sampling the vertices of a polytope is still hard even when the branchwidth is bounded

Lorenzo Najt

June 22, 2021

Abstract

We consider the complexity of sampling vertices of a polytope. A theorem of Khachiyan [27] uses the circulation polytope of a directed graph to show that this sampling problem is NP-hard, in the sense that a polynomial time sampler would imply NP = RP. It is known, also by work of Khachiyan et al. [28], that the vertex enumeration problem is NP-hard for polyhedra, while it remains open for polytopes. However, bounding the *branchwidth* has been shown to provide a total polynomial time algorithm for the polytope vertex enumeration problem, and it is therefore natural to ask whether bounding branchwidth makes vertex sampling tractable. We investigate this question and demonstrate the NP-hardness of uniformly sampling vertices of a polytope given by $\{Ax = b, x \geq 0\}$, where A has branchwidth ≤ 4. To do so, we develop gadgets that build bounded branchwidth polytopes that have many vertices over certificates of an NP-hard problem. In an appendix, we apply this gadget to provide an alternative proof a recent theorem of Guo and Jerrum about sampling vertices from another class of polytopes. We also study some related questions, such as the branchwidth of the circulation polytope, and show that the vertices of a circulation polytope of bounded branchwidth can be sampled efficiently.

1 Introduction

Polyhedra arise naturally in computer science as the set of feasible solutions of a linear program, that is, as the set of points satisfying some collection of linear inequalities. Bounded polyhedra are called polytopes. The extremal points of a polytope, also known as its vertices, provide an alternative representation of the feasible set, since every point in the feasible set is a convex combination of some of the vertices. In general, one can pass algorithmically from a description of a polytope via inequalities to the description via the set of vertices, but process is computationally very expensive as the set of vertices can be very large compared to the set of inequalities. Despite this, certain questions about the set of vertices can be answered efficiently; for instance, the ellipsoid algorithm allows one to find a vertex in polynomial time, provided one exists.

Faced with a large set of implicitly defined objects, some options for understanding it better include uniformly sampling elements, or enumerating the set as efficiently as possible. The vertex uniform sampling problem is known to be NP-hard (see [27] and appendix F), in the sense a polynomial time uniform sampler would imply RP = NP. Since the number of vertices can be exponential in the number of constraints, the best one can hope for regarding enumeration is an algorithm that spends polynomial time in the size of both the input and the output, often called a total polynomial time algorithm. It has been shown that a total polynomial time enumeration algorithm for the vertices of a polyhedron would imply P = NP [28], however, whether there is a total polynomial time vertex enumeration algorithm in the case of polytopes (bounded polyhedra) remains an open question.

On the other hand, when a parameter of the polytope called the branchwidth (§6) is bounded, the vertices can be enumerated in total polynomial time [35]. This indicates that there is some chance that bounding the branchwidth simplifies computational problems about the set of vertices. The vertex uniform sampling problem, for polytopes of bounded branchwidth, has not been studied in the literature. Due to the success of branchwidth in graph algorithms [9] and the result in [35] one can reasonably ask: is there a polynomial time algorithm for uniformly sampling the vertices of a polytope of bounded branchwidth? We answer this question in the negative, and show in Theorem 8.0.3 that uniformly sampling the vertices remains

Family of objects	Vertex Enumeration	Vertex Sampling
Polyhedra	NP-hard [28]	NP-hard [27]
Polytopes	Open Problem	NP-hard [27]
Polyhedra of bounded branch-width	Open Problem	NP-hard (This Paper)
Polytopes of bounded branch-width	Total polynomial time algorithm time [35] (The runtime exponent depends on branchwidth, and the existence of a fixed parameter tractable algorithm remains an open problem.)	NP-hard (This Paper)
Non-degenerate polyhedra	Total polynomial time algorithm [5]	Open Problem?
$\{0, \frac{1}{2}, 1\}$ polytopes	Open Problem	NP-hard [18]
$\{0, 1\}$ polytopes	Open Problem	Open Problem [18]

Figure 1: The complexity of enumeration and sampling problems, for polyhedra of various kinds. We use NP-hardness loosely, here it just means that a polynomial time solution to the problem collapses NP to a class of efficiently solvable problems such as RP or P. We note that by arguments as in appendix C counting and sampling are essentially equivalent, despite the lack of self-reducibility.

intractable even when the branchwidth of the polytope is bounded. Figure 1 summarizes the complexity of vertex enumeration and sampling for various kinds of polyhedra. Our proof relies on a result about the "restricted vertex problem" [14] from [35]. The main tool is a way to construct lifts of a polytope by exploding certain vertices into hypercubes, which we call the hypercube gadget (§5).

Our proof is fundamentally distinct from the argument in [27], which relies on circulation polytopes. Indeed, as we explain detail in the appendix, bounding the branchwidth of a circulation polytope makes the vertex sampling problem polynomial time tractable. Additionally we show in appendix B that the natural p-relation for this problem is not self reducible, unless P = NP, and examine in appendix F the branchwidth of the circulation polytope used in Khachiyan's proof, showing that in the bounded branchwidth case, vertices can be sampled from the circulation polytope. In appendix A, we give an alternative proof via the hypercube gadget of a theorem about sampling the vertices of a $\{0, \frac{1}{2}, 1\}$-polytope from [18].

2 Related Work

In [14], it was shown that for any graph $G = (V, E)$ and $s, t \in V$ a polyhedron can be constructed whose vertices correspond to the simple s, t-paths in G. Using techniques of [25, Proposition 5.1], one can show that a polynomial time algorithm for uniformly sampling simple s, t-paths does not exist unless RP = NP. Taken together, these imply an intractability result for uniformly sampling from the vertices of a polyhedron. Using the circulation polytope instead, one can show that efficiently sampling vertices of polytopes, i.e. bounded polyhedra, would imply RP = NP [27]. Since a collapse like RP = NP would be very unlikely, given that we believe P $\neq$ NP and P = RP = BPP, this is generally taken as evidence that these sampling problems are intractable. We will informally describe sampling problems as intractable throughout, with the understanding that this means that an efficient algorithm for approximate sampling would imply RP = NP.

There are intermediate complexity classes for hardness of sampling, such as relating the counting problem to #BIS; [18] relates counting vertices of a certain class of polytopes defined by network matrices to #BIS. In addition, motivated by the problem of sampling the vertices of a 0/1 polytope, [18] shows that sampling the vertices from polytopes promised to have vertices in the set $\{0, \frac{1}{2}, 1\}$ is NP-hard. We provide an alternative,

although closely related, route to their theorem in appendix A.

There are certain situations when the sampling problem can be done in polynomial time, such as if the rank or dimension is fixed.[1] Additionally, for certain polytopes there are efficient algorithms for approximately uniformly sampling or counting the vertices. For instance, this is so for the bipartite matching polytope by [24]. Other examples include: [3, 6, 7, 12, 22]. The aforementioned [18] provides some more general settings under which sampling vertices can be reduced to counting integer solutions of network flow. For some discussion of heuristic Monte-Carlo methods for counting vertices of polytopes, see [37].

A related problem is whether the random walk on the graph of the polytope mixes rapidly [26];. However, since every polytope can arise as the vertex figure of another polytope, simply by taking a cone over the polytope, the problem of implementing this random walk in general is just as hard as uniformly sampling vertices of polytopes. Thus, the theoretical interest in this random walk provides another reason to be interested in questions about uniformly sampling vertices.

Branchwidth of polytopes is a parameter that has made certain NP-hard problems on polytopes tractable. Besides the vertex enumeration problem in [35], which we discussed in the introduction, [8] showed that integer programming with bounded branchwidth non-negative constraint matrices could be solved in pseudopolynomial time, although the problem becomes NP-hard again when the signs of the entries in the constraint matrix are unrestricted. They also show that the problem is pseudopolynomial time if in addition to the constraint matrix, the variables are bounded inside of a box of the form $[0, d]^n$. The empirical investigation in [30] examines the practical usefulness of branchwidth as a parameter for solving integer programming problems, and finds that certain situations it improved on black-box integer programming solvers.

Regarding [35] in more detail, we emphasize that their result is for enumerating the vertices of a polytope of bounded branchwidth, not a polyhedron. They leave open the possibility that total polynomial time vertex enumeration causes the collapse P = NP, as in [28], for polyhedra of bounded branchwidth. The topic of vertex enumeration is motivated in [35] by applications to computational systems biology [2, 15, 40]. Finally, as mentioned in [35], for non-degenerate polyhedra vertices can be enumerated in total polynomial time [5, 10]. To the best of the authors knowledge, vertex uniform sampling and vertex counting have not been studied in the literature for non-degenerate polyhedra.

3 Main Steps for Proving Hardness of Sampling

In this section we sketch out the main steps of the hardness of sampling result. We follow the general strategy of Proposition 5.1 in [25], which has been used to prove hardness of sampling in [18, 27, 34] among many other places. First, recall:

Definition 3.0.1 (RP)**.** *RP is the class of decisions problems for which there is a polynomial time randomized algorithm that always answers NO if a instances is a NO-instance, and, if the instance is a YES-instance, answers YES with probability* $\geq 3/4$.

The main steps of the argument are as follows:

1. We identify an NP-hard optimization problem about the vertices of a polytope. For a given polytope, suppose that $f : \mathrm{Vert}(P) \to \mathbb{N}$ is the function we wish to maximize.
2. We find a way to efficiently build a family of associated polytopes P^d, so that there is a map $\pi^d : P^d \to P$, which restricts to a surjection on the vertices and has the property that $|(\pi^d)^{-1}(x) \cap \mathrm{Vert}(P^d)| = \Theta(2^{f(x)d})$.
3. This means that if we draw a uniform vertex of P^d, and apply π^d to it, we end up with a random vertex of P whose distribution is increasingly concentrated on the vertices maximizing f. In particular, we design our construction so that the number of vertices above a vertex x grows exponentially in d at a rate that depends on $f(x)$.

[1] It is known [33] that a polytope of dimension d with m facets has $O(m^{d/2})$ vertices. Moreover, the enumeration problem for fixed dimension can be solved in polynomial time [10]. Thus, uniform sampling can be achieved in polynomial time if the dimension of the polytope $\{Ax \leq b\}$ is fixed. Similarly, if the rank r of the matrix A in $\{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ is fixed, then the number of basic feasible solutions is $O(n^r)$, so uniform sampling can again be achieved in polynomial time by simple rejection sampling or enumeration. More refined questions about the complexity of counting and sampling with regard to these parameters are open, to the best of the authors knowledge.

4. We then take d large enough so that the potentially large number of vertices $x \in \mathrm{Vert}(P)$ with $f(x) < \max f$ is insignificant compared to the effect of the faster rate of growth. More precisely, if $S \subseteq \mathrm{Vert}(P)$ is the set of vertices maximizing f, we ensure that $(\pi^d)^{-1}(S^c) \cap \mathrm{Vert}(P^d) \gg (\pi^d)^{-1}(S) \cap \mathrm{Vert}(P^d)$
5. Thus, we can use a uniform sampler to provide an RP algorithm solving the decision version of that NP-hard optimization problem. That is, for given P and k the algorithm builds P^d, for some explicitly calculated d, samples a vertex x from P^d, and returns whether $f(\pi^d(x)) \geq k$. This is an RP algorithm because it never finds an x with $f(\pi^d(x)) \geq k$ when no such x can exist, and if there is such a $\pi^d(x)$ with $f(\pi^d(x)) \geq k$, d has be tuned so that one is produced with probability $\geq 3/4$.

The NP-hard problem we use is the restricted vertex problem, covered in §7. The construction of the P^d is done by the hypercube gadget (§5). The main technical step is ensuring that π^d restricts to a surjection on the vertices, and counting $|\pi^d(x) \cap \mathrm{Vert}(P^d)|$, which we cover in corollary 5.0.3.

4 Notation and Background

We establish the notation and conventions used in this paper, and state a few useful facts about the combinatorics of polyhedra and polytopes. Throughout, n will refer to the ambient dimension of the polyhedron.

Definition 4.0.1 (Equational form)**.** *Define $P(A,b) := \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$. This presentation of a polyhedron is called equational form.*

Definition 4.0.2. *A polyhedron always refers to the tuple consisting of the set of points in a polyhedron $P(A,b)$ defined in equational form , and the constraints A and b: $(P(A,b) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}, A, b)$.*

Definition 4.0.3 (Polytope)**.** *A polytope is a polyhedron $(P(A,b), A, b)$ for which $P(A,b)$ is a bounded set.*

When we refer to "a polyhedron" or "a polytope" (bounded polyhedra) in this paper, we are implicitly referring to a presentation of a polyhedron in equational form. This is important both for representing polyhedron in a format that can be considered the input to algorithms, and for defining the notion of branchwidth; a parameter of the algorithmic "complexity" of the polyhedra that has more to do with the interactions between defining equations than it does with its intrinsic geometry. We will recall its definition later.

The main object of study in this paper is the following:

Definition 4.0.4 (Vertices of a polytope)**.** *Let $x \in P(A,b)$. Then x is said to be a vertex of P if there is some affine hyperplane H so that $H \cap P(A,b) = \{x\}$. The set of vertices of a polytope P is denoted by* $\mathrm{Vert}(P)$.

Later on, it will be convenient to identify vertices with the coordinates where they are zero.

Definition 4.0.5 (Support of a vertex)**.** *Let $v \in \mathrm{Vert}(P(A,b))$. The zeros of v is the set $\{i \in [n] : v_i = 0\}$, and the support of v is the set* $\mathrm{supp}(v) = \{i \in [n] : v_i \neq 0\}$.

Definition 4.0.6 (Restriction of a vector or matrix)**.** *We think of $x \in \mathbb{R}^n$ as a function on $[n]$, when convenient. So, if $K \subseteq [n]$, for $x \in \mathbb{R}^n$ we let $x_K \in \mathbb{R}^K$ be the restriction of the variable. We think of this as a row vector with $|K|$ entries. Similarly, if $A \in \mathbb{R}^m \times \mathbb{R}^n$, we think of it as a $\mathbb{R}^m$ valued function on $[n]$, and denote by A_K the restriction. Equivalently, this is the matrix made up of the columns of A in K. Concretely, let $A = \begin{pmatrix} a_1 & \dots & a_n \end{pmatrix}$ be a matrix, with columns $a_1, \dots, a_n$ indexed by $[n]$. For $I = \{i_1, \dots, i_m\} \subseteq [n]$, we define $A_I = \begin{pmatrix} a_{i_1} & \dots & a_{i_m} \end{pmatrix}$.*

We will need the following:

Lemma 4.0.7 (Lemma 4.2.1 in [31])**.** *A point $x \in P = P(A,b)$ is a vertex iff the columns of the matrix A_K are linearly independent, where $K = \mathrm{supp}(x)$.*

This has the following corollary:

Corollary 4.0.8. *If x, x' are vertices of $P = P(A, b)$, and $\text{supp}(x) = \text{supp}(x')$, then $x = x'$.*

Proof. Let $K = \text{supp}(x) = \text{supp}(x')$. Then A_K has linearly independent columns by lemma 4.0.7.. Moreover, $b = Ax = A_K x_K + A_{K^c} x_{K^c} = A_K x_K$ as $x_{K^c} = 0$. Similarly $b = A_K x'_K$. Thus, $A_K(x_K - x'_K) = 0$. Since A_K has linearly independent columns, $x_K = x'_K$, thus $x = x'$. □

We also use the following vocabulary:

Definition 4.0.9 (Lift of a polytope)**.** *Let $P(A, b)$ be a polyhedron in $\mathbb{R}^n$. A polyhedron Q in $\mathbb{R}^n \times \mathbb{R}^m$ is said to be a lift of P if the projection onto the first n coordinates surjects onto the underlying set of P.*

5 Hypercube Gadget

In this section we introduce a gadget similar to the chain of diamonds construction used in [25] to show that uniformly sampling directed simple cycles is NP-hard. That result of [25] was then used in [27] to prove that sampling a polytopes vertices is NP-hard. However, as discussed in appendix F, the circulation polytope based construction of [27] cannot be used to show hardness of sampling vertices for branchwidth bounded polytopes, since the polytope branchwidth bound translates into a graph branchwidth bound, which renders the problem tractable. Therefore, in some sense, our argument abstracts away the main idea from the circulation polytope argument into a gadget that can be applied directly to the polytope, and we are able to prove a stronger theorem as a result.

This gadget takes a polytope P, defined in equational form $P = P(A, b) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$, along with a subsets $I \subseteq [n]$, and builds a family of polytopes $P_{d,I}$, which are lifts of a polytope contained in P. These lifts will be designed to have exponentially more vertices above the vertices of P that are the certificates to an NP-hard problem defined in terms of (A, b, I). That NP-hard problem will be discussed in §7.

Definition 5.0.1 (Hypercube Gadget)**.** *Let $I \subseteq [n]$, and $d \geq 1$. Suppose P is given by $P = P(A, b) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$. Define a polytope $P^{I,d}$ by introducing variables $y_{i,j}$ and $y^s_{i,j}$ for $j = 1, \ldots, d$ and each $i \in I$, with the constraints $y_{i,j}, y^s_{i,j} \geq 0$ and $y_{i,j} + y^s_{i,j} = x_i$ for each $j = 1, \ldots, d$, $i \in I$. That is $P^{I,d} = \{(x, y, y^s) \in \mathbb{R}^n \times (\mathbb{R}^d)^{|I|} \times (\mathbb{R}^d)^{|I|} : Ax = b, (x, y, y^s) \geq 0, y_{i,j} + y^s_{i,j} = x_i, \forall j \in [d], i \in I\}$. We let $A^{I,d}$ denote the matrix defining $P^{I,d}$, so $P^{I,d} = P(A^{I,d}, [b_1, \ldots, b_n, 0, \ldots, 0])$. Define a map $\pi^{I,d} : P^{I,d} \to P$ by remembering only the x coordinates, that is $\pi^{I,d}(x, y, y^s) = x$. When clear from context we refer to this map as π.*

The conditions on the y variables are such that for fixed x, the y variables can take on values within a hypercube: after eliminating slack variables and rewriting the $y_{ik} + y^s_{ik} = x_i$ as an inequality gives $0 \leq y_{ik} \leq x_i$ for $k = 1, \ldots, d$ and $i \in I$. Thus, the intuition is that we have created a new polytope which has hypercube of dimension $d|\text{supp}(x) \cap I|$ above a vertex $x \in \text{Vert}(P)$. Moreover, as we verify shortly, the vertices of those hypercube fibers account exactly for all the vertices of the lift. See fig. 2 for some depictions of this construction. The following lemma verifies this intuition and characterizes the vertices of the hypercube gadget.

Lemma 5.0.2.

$$\text{Vert}(P^{I,d}) = \bigcup_{x \in \text{Vert}(P)} \bigcup_{y \in \text{Vert}(H_x)} \{(x, y)\}$$

where $H_x = \{(y, y^s) : y, y^s \geq 0, y_{i,j} + y^s_{i,j} = x_i\}$ is the hypercube in $P^{I,d}$ above x.

Proof. To save on notation, we will write $P^{I,d}$ without the slack variables. So, let $\tilde{P} = \{(x, y) \in \mathbb{R}^n \times (\mathbb{R}^d)^{|I|} : ((x, y) \geq 0) \wedge (Ax = b) \wedge (\forall i \in I, j \in [d], (y_{i,j} \leq x_i))\}$, and let $\pi(x, y) = x$. First, we show that $\pi(\text{Vert}(\tilde{P})) \subseteq \text{Vert}(P)$, and then we will use that to finish the proof of the lemma.. Throughout we will use the following characterization of vertices of a polytope: a point $x \in P$ is a vertex of P iff there do not exist $x', x'' \in P$, $x' \neq x''$ and $\lambda \in (0, 1)$ with $x = \lambda x' + (1 - \lambda)x''$.

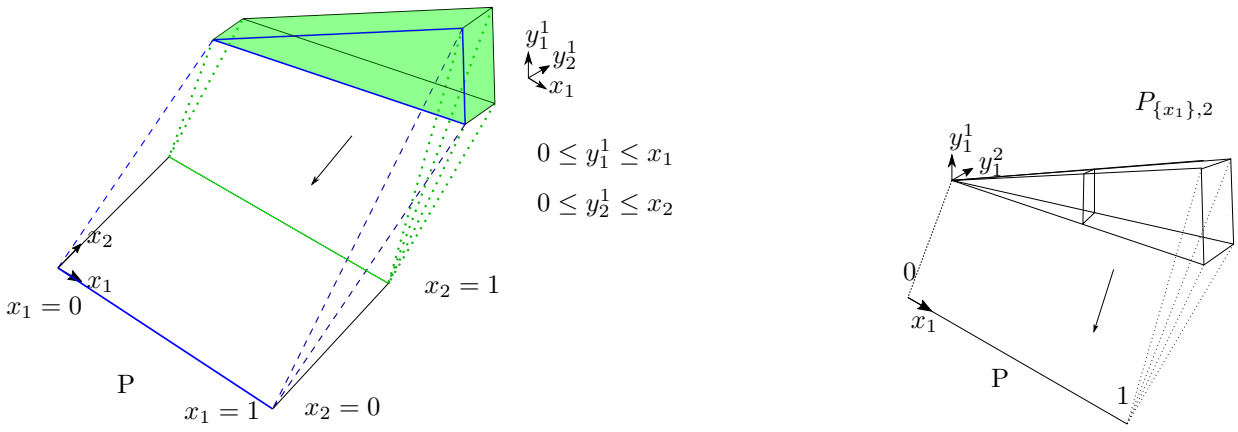

Figure 2: Two examples of the hypercube gadget, albeit presented in $Ax \le b$ form for sake of visualization. On the right we depict an instance of the hypercube gadget applied to $P = \{0 \le x_1 \le 1\}$, with $d = 2$ and $I = \{1\}$. The entire $1+2 = 3$ dimensional polytope created by the gadget is drawn, and the arrow shows the projection map π. The left hand picture depicts two fibers of the hypercube gadget with $I = \{1, 2\}$ above the polytope given by $0 \le x_1, x_2 \le 1$. The additional variables of the lift are y_1^1, y_2^1. Since the polytope is $2 + 2 = 4$ dimensional, we cannot draw the entire thing, and instead we draw fibers of the projection map. The fiber $\pi^{-1}(\{x_2 = 0\})$ is in blue and $\pi^{-1}(\{x_2 = 1\})$ is in green. Note that the two fibers are disjoint in $P^{I,1}$, but as we only draw the x_1, y_1^1, y_2^1 axis for the lift, we cannot see that.

1. Proof of $\pi(\mathrm{Vert}(\tilde{P})) \subseteq \mathrm{Vert}(P)$: Suppose that $(x, y) \in \mathrm{Vert}(\tilde{P})$ with $\pi(v) \notin \mathrm{Vert}(P)$. Then, there are $x' \neq x'' \in P$ and $\lambda \in (0, 1)$ such that $\pi(v) = \lambda x' + (1 - \lambda)x''$. Moreover, we have that $y_{ik} \le x_i$ for each $i \in I$ and $k \in [d]$. Therefore, we have that for each $i \in I$, there are some $\alpha_{ik} \in [0, 1]$ with $y_{ik} = \alpha_{ik} x_i$. We define $y' \in V_y$ by $y'_{ik} = \alpha_{ik} x'_i$ and y'' by $y''_{ik} = \alpha_{ik} x'_i$. We then consider the points $v' = (x', y')$ and $v'' = (x'', y'')$. We observe that $v', v'' \in \tilde{P}$ since the y coordinates were designed to satisfy the defining condition: $y'_{ik} = \alpha_{ik} x'_i \le x'_i$, and likewise for the y''. We now obtain a contradiction by verifying that $v = \lambda v' + (1 - \lambda)v''$: s this is clear for the x coordinates, we only need to observe that $\lambda y'_{ik} + (1 - \lambda)y''_{ik} = \lambda \alpha_{ik} x'_i + (1 - \lambda)\alpha_{ik} x''_i = \alpha_{ik}(\lambda x'_i + (1 - \lambda)x''_i) = \alpha_{ik} x_i = y_{ik}$.
2. Conclusion of proof: First, we show that $\mathrm{Vert}(\tilde{P}) \subseteq \bigcup_{x \in \mathrm{Vert}(P)} \bigcup_{y \in \mathrm{Vert}(H_x)} \{(x, y)\}$. Suppose that $(x, y) \in \mathrm{Vert}(\tilde{P})$ but y is not a vertex of H_x, where $H_x = \{y_{ik}, i \in I, k = 1, \ldots d : y \ge 0, y_{ik} \le x_i\}$. Then there are $y' \neq y'' \in H_x$ and $\lambda \in (0, 1)$ so that $y = \lambda y' + (1 - \lambda)y''$. Thus, we have $(x, y'), (x, y'') \in \tilde{P}$ with $(x, y) = \lambda(x, y') + (1 - \lambda)(x, y'')$, contradicting the assumption that $(x, y) \in \mathrm{Vert}(\tilde{P})$. Since we already know that $x \in \mathrm{Vert}(P)$ from the previous bullet, the inclusion follows. We next show the other inclusion, namely that $\mathrm{Vert}(\tilde{P}) \supseteq \bigcup_{x \in \mathrm{Vert}(P)} \bigcup_{y \in \mathrm{Vert}(H_x)} \{(x, y)\}$. Let $x \in \mathrm{Vert}(P)$ and $y \in \mathrm{Vert}(H_x)$. For contradiction, suppose that such an $(x, y) \notin \mathrm{Vert}(\tilde{P})$ is not a vertex. Then we have $(x', y') \neq (x'', y'') \in \tilde{P}$ and $\lambda \in (0, 1)$ with $\lambda(x', y') + (1 - \lambda)(x'', y'') = (x, y)$. If $x' \neq x''$, then x was not a vertex of P, a contradiction, since we showed in the first part that the vertices of $\tilde{P}$ map to the vertices of P. If $x' = x''$ but $(x', y') \neq (y', y'')$ we must have $y' \neq y''$, meaning that $y \notin \mathrm{Vert}(H_x)$, which contradicts our assumption.

□

This immediately gives the following corollary:

Corollary 5.0.3. *With notation as in definition 5.0.1, We have that:*

1. $\mathrm{Vert}(P^{I,d}) \subseteq \pi^{-1}(\mathrm{Vert}(P))$
2. *For* $x \in \mathrm{Vert}(P)$, $|\pi^{-1}(x) \cap \mathrm{Vert}(P^{I,d})| = 2^{d|\mathrm{supp}(x) \cap I|}$.

Remark 5.0.4. *TODO: Relate to 1-flux modules from [35]?*

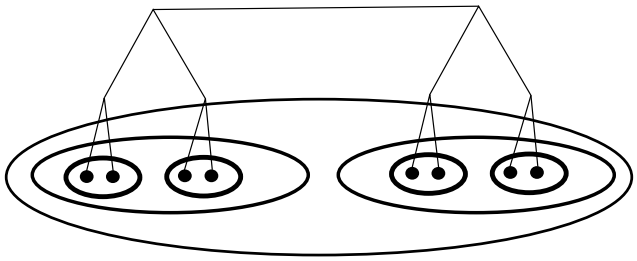

Figure 3: The hierarchical decomposition of a set X given by an X-labelled cubic tree. The dots represent the 8 elements of X, and the circles represent the blocks.

6 Branchwidth

In this section, we examine how the branchwidth of a polytope changes under the hypercube gadget. In this section we give the definition of branchwidth, which we break into a few subdefinitions in order to make later arguments easier to state precisely.

As motivation for the concept of branchwidth, we recall that one of the common features of NP-hard problems are variables with complex interactions between different choices of their settings. Some approaches in fixed parameter tractability try to limit interaction between variables, and hope that the limited interactions make certain decision problems simpler. In the linear programming setting, our variables are non-negative real numbers $(x_1, \ldots, x_n) = x$, and our constraints are given by the equation $Ax = b$. Branchwidth will control the interaction between variables by means of a tree whose leaves are labelled by columns of a matrix A. The tree can be thought of as a hierarchical two-partitioning of the set of columns, where the interaction between the blocks of the 2-partitions is what we intend to control. The measurement of interaction between two sets of columns will related to the dimension of the intersection of their spans. We now explain in detail.

Definition 6.0.1 (X-labelled Cubic Trees)**.** *A tree T with nodes of degree 3 or 1 is called a cubic tree. The degree 1 nodes of T are also called its leaves, leaves(T). An X-labelling ν of a cubic tree T is a bijection between* leaves(T) *and a set X, $\nu : \text{leaves}(T) \to X$. An X-labelled cubic tree is a pair consisting of a cubic tree T and a labelling ν of its leaves (T, ν).*

An X-labelled cubic tree provides a family of 2-partitions of X:

Definition 6.0.2 (Set of partitions induced by an X-labelled tree)**.** *Consider an X-labelled tree (T, ν). For each $e \in E(T)$, let $P_\nu(e) = \{L_e, R_e\}$ be the 2-partition of X with blocks given by the ν labels of the leaves of the two connected components of $T \setminus e$. That is, $P_\nu(e) = \{\{\nu(v) : \text{ for } v \in \text{leaves}(Y)\} \text{ for } Y \in \text{ConnectedComponents}(T \setminus e)\}$. We also define the set of partitions of X induced by (T, ν) as Partitions$((T, \nu)) := \{P_\nu(e) : e \in E(T)\}$*

Remark 6.0.3 (Pedagogical remark regarding branchwidth.)**.** *It is useful to recognize that an X labelled tree produces a hierarchical family of two partitions of X. That is, let T be an X-labelled tree, and consider any edge $e = \{a, b\}$ in T, with corresponding 2-partitions of X, $\{A, B\}$, where we have chosen the labels so that a is in the connected component, say T_a, of $T \setminus e$ whose leaves correspond to A. Then, T_a gives us an A-labelled cubic tree, from which we can again obtain various 2-partitions. See fig. 3 for an illustration of this.*

From an algorithmic perspective, these hierarchical 2-partitions give a framework for obtaining a solution to a problem by gluing together solutions to subproblems, in the way that is usually done in dynamic programming. The paper [8] applies this idea to integer programming, for instance, and many algorithms based on branch or tree decompositions of graphs follow a similar strategy.

We now specialize X-labelled trees to the set of columns of a matrix to arrive at the notion of a branch decomposition:

Definition 6.0.4 (Branch Decomposition)**.** *Let A be a matrix, and let $[n]$ index the columns of A. A branch decomposition of A is an $[n]$-labelled cubic tree (T, ν). We let $BD(A)$ denote the set of branch decompositions of A.*

Thus, we have established a way to hierarchically decompose the columns of a matrix A. Next, we turn to measuring the interaction between complementary pieces.

Definition 6.0.5 (Width and the Connectivity Function of a Branch Decomposition)**.** *Let (T, ν) be a branch decomposition of A. The* width *of the edge e with $\{L_e, R_e\} = P_\nu(e)$ is defined to be $\lambda(e) := \mathrm{rk}(A_{L_e}) + \mathrm{rk}(A_{R_e}) - \mathrm{rk}(A) + 1 = \dim(\mathrm{Span}(A_{L_e}) \cap \mathrm{Span}(A_{R_e})) + 1$. The function $\lambda_{(T,\nu)} : E(T) \to \mathbb{N}$ is called the* connectivity function *of the branch decomposition. The width of the branch decomposition is $\max_{e \in E(T)} \lambda_{(T,\nu)}(e))$.*

Finally, we can define the branchwidth of a matrix:

Definition 6.0.6 (Branchwidth)**.** *The* branchwidth *of a matrix A, $bw(A)$, is the minimum width of any branch decomposition of A. That is,*

$$\mathrm{bw}(A) = \min_{(T,\nu) \in BD(A)} (\max_{e \in E(T)} \lambda_{(T,\nu)}(e))$$

Remark 6.0.7. *There are two conventions for branchwidth in the literature, which differ by 1. [8, 36] uses $\lambda(e) = r(L_e) + r(R_e) - r(A) + 1$, and [35] uses $\lambda(e) = r(L_e) + r(R_e) - r(A)$. To comport with the majority of the literature, we follow the first convention. Since we use a result from [35] for our reduction, this will require a slight translation when we get to it.*

We now define notion of branchwidth for polyhedra defined in equation form. Note that this definition makes explicit reference to the system of equations defining A, and is not just a measurement of the geometry of the convex set of points making up the polytope.

Definition 6.0.8 (Branchwidth of a polytope)**.** *Let $P = (P(A, b), A, b))$ be a polytope. Then branchwidth of P is* $\mathrm{bw}(A)$.

As mentioned previously, polytopes of bounded branchwidth are *sometimes* more amenable to computation. One can see [8] and [35] for examples. However, in contrast to the similar notion of *graph* branchwidth or treewidth, the usefulness of branchwidth appears very limited. In particular, we will show that the vertex sampling problem remains hard even when the branchwidth is 4.

6.1 Branchwidth Lemmas

In this section we prove several lemmas about branchwidth that we will use repeatedly in our calculations. In particular, we use these to calculate the affect of the hypercube gadget on branchwidth.

Lemma 6.1.1. *Let A be any matrix, and let $A' = \begin{pmatrix} A & v \end{pmatrix}$, where v is any column vector of the same dimension as any other column of A. Then* $\mathrm{bw}(A) \leq \mathrm{bw}(A')$.

Proof. Let T' be an optimal tree decomposition for A'. Let T be obtained by deleting the leaf corresponding to n, the last column of A', and smoothing its parent node. For each partition of $[n+1]$, (L, R), obtained from T', assume without loss that $n \in R$, so that the corresponding partition of $[n]$ obtained from T is $(L, R \setminus n)$. Since $\mathrm{Span}(A_L) \cap \mathrm{Span}(A_{R \setminus n}) \subseteq \mathrm{Span}(A'_L) \cap \mathrm{Span}(A'_R)$, the claim follows. □

Lemma 6.1.2. *If S is a permutation matrix, then* $\mathrm{bw}(AS) = A$.

Proof. Note that the affect of right multiplication by S is to permute the columns of A. Thus, we can use the permutation matrix S to translate any branch decomposition of A into one of AS, without changing the width, and vica-versa. □

The following definition will be useful for describing calculations with branchwidth.

Definition 6.1.3 (Splitting a leaf)**.** *Let (T, ν) be an $[n]$-labelled cubic tree, and let $i \in [n]$. Let L Be the leaves of T, and let $v = \nu^{-1}(i) \in L$. Let T' be the tree obtained by adding two leaf nodes, a and b, connected to the node $v = \nu^{-1}(i)$ of T, and let ν' be the labelling of the leaves of T' such that $\nu'|_{L \setminus v} = \nu|_{L \setminus v}$ and $\nu'(a) = i$, $\nu'(b) = n + 1$. Then (T', ν') is an $[n+1]$-labelled cubic tree, which we refer to as the splitting of T at the leaf i. See fig. 4.*

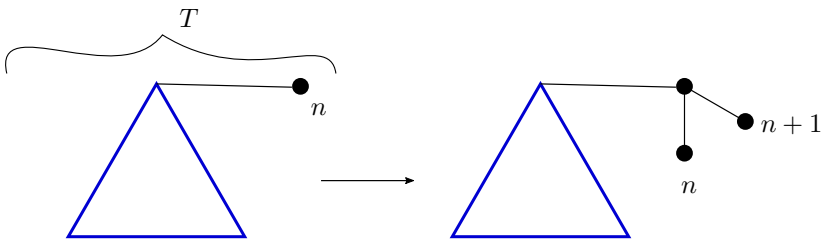

Figure 4: An illustration of splitting the leaf n into two leaves labelled n and $n+1$.

Splitting a leaf changes the set of partitions in the following way:

Lemma 6.1.4. *Let (T, ν) be a tree decomposition of a matrix A, and let (T', ν') be obtained by splitting T at the leaf n. The set of partitions of $[n+1]$ induced by T is given by Partitions$((T', \nu')) = \{\{L, R \cup \{n+1\}\}$ for $\{L, R\} \in$ Partitions$((T, \nu))$ s.t. $n \in R\} \cup \{\{n\}, \{1, 2, \ldots, n-1, n+1\}\} \cup \{\{n+1\}, [n]\}$.*

We define two more operations on matrices, which, together with the previous lemmas, will let us calculate the effect of the hypercube gadget on branchwidth.

Definition 6.1.5 (Parallel Extension)**.** *Let $A = (v_1, \ldots, v_n)$ be a matrix. Suppose that $A' = (v_1, \ldots, v_n, v_n)$; that is, A' is obtained from A by adding a copy of its last column. Then we say that A' is a parallel extension of A.*

Lemma 6.1.6 (Parallel Extension Lemma)**.** *Let A' be a parallel extension of A. Then,* $\mathrm{bw}(A) \leq \mathrm{bw}(A') \leq \max(2, \mathrm{bw}(A))$.

Proof. The lower bound $\mathrm{bw}(A) \leq \mathrm{bw}(A')$ follows from lemma 6.1.1, so we consider the other bound. Let (T, ν) be an optimal branch decomposition of A, and let (T', ν') be the splitting of T at the leaf n. From lemma 6.1.4 we know that Partitions$((T', \nu')) = \{\{L, R \cup \{n+1\}\}$ for $\{L, R\} \in$ Partitions$((T, \nu))$ s.t. $n \in R\} \cup \{\{n\}, \{1, 2, \ldots, n-1, n+1\}\} \cup \{\{n+1\}, [n]\}$. For the last two partitions, the dimension of the intersection cannot exceed 1, hence the corresponding width is at most 2. For the partitions of the form $\{\{L, R \cup \{n+1\}\}$ for $\{L, R\} \in$ Partitions$((T, \nu))$ s.t. $n \in R\}$ we observe that $\mathrm{Span}(A_R) = \mathrm{Span}(A_{R \cup \{n+1\}})$, because $n \in R$, and $\mathrm{Span}(A_L) = \mathrm{Span}(A'_L)$, hence the width of the corresponding edges is unchanged. Thus, we have a branch decomposition for A' with width $\leq \max(2, \mathrm{bw}(A))$, proving that $\mathrm{bw}(A') \leq \max(2, \mathrm{bw}(A))$. □

Definition 6.1.7 (Series extension)**.** *Given a matrix A, a matrix $A' = \left(\begin{smallmatrix} A & 0 \\ v & 1 \end{smallmatrix}\right)$, such that v is zero in every entry except exactly one, will be called a series extension of A.*

Lemma 6.1.8 (Series Extension Lemma)**.** *Let $A \in \mathbb{R}^{n \times m}$, and define $A' = \left(\begin{smallmatrix} A & 0 \\ v & 1 \end{smallmatrix}\right)$ be any series extension of A. Then* $\mathrm{bw}(A) \leq \mathrm{bw}(A') \leq \max(2, \mathrm{bw}(A))$.

Proof. Assume that $[n]$ indexes the columns of A. Using lemma 6.1.2 we can assume that the coordinate where v is nonzero is the nth coordinate. Let (T', ν') be the branch decomposition for A' obtained by splitting (T, ν) at the leaf n. Again, from lemma 6.1.4 we know that Partitions$((T', \nu')) = \{\{L, R \cup \{n+1\}\}$ for $\{L, R\} \in$ Partitions$((T, \nu))$ s.t. $n \in R\} \cup \{\{n\}, \{1, 2, \ldots, n-1, n+1\}\} \cup \{\{n+1\}, [n]\}$. For the last two partitions induced by T', namely, $\{\{n+1\}, [n]\}$ and $\{\{n\}, \{1 \ldots, n-1, n+1\}\}$, we note that the dimension of the span of the intersections can be at most 1, so the corresponding width is at most 2.

For the partitions induced by (T', ν') of the form $\{L, R \cup \{n+1\}\}$ for $\{L, R\} \in$ Partitions$((T, \nu))$ with $n \in R$, we first observe that $i(\mathrm{Span}(A_L)) = \mathrm{Span}(A'_L)$ where $i : \mathbb{R}^m \to \mathbb{R}^{m+1}$ is given by $i(v) = (v, 0)$, since $v|L = 0$. Next, we observe that $\mathrm{Span}(A'_{R \cup \{n+1\}}) = \mathrm{Span}(i(\mathrm{Span}(A|R)), e_{m+1}))$. Thus, $\mathrm{Span}(A'_L) \cap \mathrm{Span}(A'_{R \cup \{n+1\}}) = i(\mathrm{Span}(A_L) \cap i(\mathrm{Span}(A_R)) = i(\mathrm{Span}(A_L) \cap \mathrm{Span}(A_R))$, as i is injective, which means the edges of T have the same width in both T and T'. In addition to bounding the width of the edges of T' not already covered, this also implies the lemma's claim that $\mathrm{bw}(A) \leq \mathrm{bw}(A')$.

Thus, putting these two calculations together, we see that the width of the branch decomposition (T', ν') is at most $\max(2, \mathrm{bw}(A))$, which implies $\mathrm{bw}(A') \leq \max(2, \mathrm{bw}(A))$.

□

Remark 6.1.9. *The reason for the terminology used in the parallel extension is that the last two columns of A' are parallel elements of the matric matroid. The reason for the series terminology is that any basis of the matric matroid A' contains at least one of i and $n+1$, where i is the column where v in the definition is nonzero. Hence, the new element $n+1$ is in series with the element i.*

6.2 Hypercube Gadget and Branchwidth

We bound the influence of the hypercube gadget on branchwidth by showing that it is a sequence of series-parallel extensions, and then applying the lemmas of the previous section.

Lemma 6.2.1. *Suppose that the polytope P is defined by $\{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$. Let $P^{I,d} = P'$ be given in equational form by $P(A', b')$. Then* $\mathrm{bw}(A') = \max(\mathrm{bw}(A), 2)$.

Proof. First, note that introducing a pair of variables $y_{i,j}, y^s_{i,j}$ with the equation $y_{i,j} + y^s_{i,j} = x_i$ changes the defining matrix A by adding two new columns, and then a row vector that is -1 in the ith column, and that ends with two 1s and is zero everywhere else. That is: $A' = \left(\begin{smallmatrix} A & 0 & 0 \\ v & 1 & 1 \end{smallmatrix}\right)$ where $v = -\delta_i$. This is a series extension, followed by a parallel extension: $A \to \left(\begin{smallmatrix} A & 0 \\ -\delta_i & 1 \end{smallmatrix}\right) \to \left(\begin{smallmatrix} A & 0 & 0 \\ -\delta_i & 1 & 1 \end{smallmatrix}\right)$. The constraint matrix, $A^{I,d}$, defining the hypercube gadget is in turn obtained by repeating this sequence of series then parallel extensions for each $i \in I$ and for each $j = 1, \ldots d$. The claim now follows by Lemma 6.1.6 and Lemma 6.1.8. □

Remark 6.2.2. *This perspective shows that the hypercube gadget, being a sequence of series and parallel extensions applied to the set of variables of interest, is closely related to the chain of diamonds gadget used in Proposition 5.1 of [25].*

7 Restricted Vertex Problem

Both theorems of this paper are reductions from the following problem:

Restricted Vertex Problem (RVP)
Input: An integer $m \times n$ matrix A, a vector $b \in \mathbb{Z}^m$ and $I \subseteq [n]$.
Output: Is there a *vertex* x of $\{Ax = b, x \geq 0\}$ such that $x_i > 0$ for all $i \in I$?

In other words, this problem asks whether there is a *vertex* x of $P(A, b)$ with support containing I.

Remark 7.0.1 (See [14])**.** *We note that if problem asked instead whether there is a point $x \in P(A, b)$, not necessarily a vertex, with $x_i > 0$, then the problem can be solved by the following linear program:*

$$\begin{aligned} &\max(m) \\ &(x, m) \in \mathbb{R}^n_{\geq 0} \times \mathbb{R}_{\geq 0} \\ &Ax = b \\ &m \leq x_i, i \in I \end{aligned}$$

This linear program works because if there is a point x with $x_i > 0$ for all $i \in I$, then the optimum value of m is > 0. Otherwise, the optimum value is 0. The difficulty in the restricted vertex problem is thus that it requires a vertex of $P(A, b)$ with the prescribed property, not just any point. In addition, note that the vertices of this linear program do not necessarily map down to the vertices of $P(A, b)$, as in general the optimal solutions will occur in the interior of some face; for an explicit example, one can take $x_1 + x_2 = 1, I = \{1, 2\}$.

Work in [14] showed the restricted vertex problem to be strongly NP-complete on polytopes. They investigated this in the context of analyzing backtracking algorithms for vertex enumeration, for which the above problem would be a key pruning subroutine. Remark 7.0.1 is from [14] and is a tool in a pruning subroutine for a face enumeration algorithm they present. In [35] the restricted vertex problem was analyzed again in the context of enumeration, but this time with branchwidth constraints. In particular, they proved:

Theorem 7.0.2. *[35][Theorem 10] RVP is NP-complete on the class of polytopes of branchwidth* ≤ 4.

Theorem 7.0.2 will be the NP-hardness result which we will use to show the hardness of sampling; we will use the hypercube gadget in order to concentrate probability on vertices that maximize $|\text{supp}(v) \cap I|$, which is the optimization problem version of the decision problem RVP.

8 Proof of Sampling Intractability

In this section we prove the theorem on intractability of sampling, following the idea introduced in [25]. To state a stronger theorem, we recall the notion of total variation:

Definition 8.0.1. *Let* P, Q *be two probability distributions on a finite set* X. *Then* $||P - Q||_{TV} = \sup_{A \subseteq X} |P(A) - Q(A)|$.

The intuition for this definition is that if $||P - Q||_{TV} \leq \epsilon$, then the probability that an event A occurs differs by at most ϵ under in P or Q. In particular, if ϵ is sufficiently small then P and Q are practically indistinguishable under all statistical tests 1_A. In these hardness of sampling settings, it is typical to be unable to sample from a distribution that is within any nontrivial constant total variation distance from the target distribution, because all we need is a constant probably of sampling from the set of certificates to the hard problem in order to get the collapse NP = RP.

We need the following lemma:

Lemma 8.0.2. *A polytope* $P(A, b) \subseteq \mathbb{R}^n$, *i.e. where* A *has* n *columns, has at most* 2^n *vertices.*

Proof. Each vertex is given by some basis of A, which is specified by a subset of the set of columns. □

Theorem 8.0.3 (Intractability of sampling vertices)**.** *Fix any* α, $0 \leq \alpha < 1$. *Suppose that there was a polynomial time probabilistic Turing machine* M *such that for each polytope* $P(A, b)$ *of branchwidth* ≤ 4, $M(A, b)$ *outputs a sample from* q_P, *where* q_P *is any probability distribution on* $\text{Vert}(P)$ *with* $||q_P - \textit{Uniform}_{\text{Vert}(P)}|| \leq \alpha$. *Then* RP = NP.

Proof. Consider an instance $((A, b), I)$ of RVP with $P(A, b)$ a polytope of branchwidth ≤ 4. Let $S \subseteq \text{Vert}(P)$ be the set of vertices solving that instance of RVP, that is, for $x \in S$ is a vertex with $\text{supp}(x) \supseteq I$, and suppose that $|S| > 0$. Construct $P^{d,I} =: P^d$, i.e. by calculating the constraints according to the definition of the hypercube gadget; then, by lemma 5.0.2, P^d has $2^{|I|d}$ vertices above any vertex of P solving the given instance of RVP. Above any other vertex, it has at most $2^{(|I|-1)d}$ vertices. Suppose now that this instance of RVP has a solution. Since $P(A, b)$ has at most 2^n vertices, P^d has at most $2^n 2^{(|I|-1)d}$ vertices which are not witnesses to the instance of RVP, and at least $2^{|I|d}$ vertices which are. If $H = \pi^{-1}(S) \cap \|(P^d)$ and $N = \text{Vert}(P^d) \setminus H$, we have $|H| \geq D|N|$ with $D = 2^{d-n}$, so $\frac{|H|}{|H|+|N|} \geq \frac{D}{1+D}$, and so a uniform sample from the vertices of P^d projects under π to a solution to RVP with probability at least $\frac{2^{d-n}}{1+2^{d-n}} \geq 1 - 2^{-m}$ for $d = n + m$. If we take m so that $1 - 2^{-m} \geq 1 - \alpha/2$, then a sample from M applied to P^d, which we can obtain in polynomial time since P^d has branchwidth ≤ 4 by lemma 6.2.1, has probability at least $\alpha/2$ of sampling from a witness to the RVP instance. If the RVP instance had no solutions, this algorithm of course never produces one. Thus, as this entire process takes polynomial time, M provides an RP algorithm for the NP-hard problem RVP, proving the claim. □

NP-hardness of the corresponding counting problem can also be shown, see appendix C.

Remark 8.0.4. *We require that* M *run correctly only on polytopes that have bounded branchwidth. In particular, one can formulate this as a promise problem, as in [18]. We do not make any claims about verifying the branchwidth bound.*

9 Acknowledgements

This paper grew out of a question and answer on the theoretical computer science stack exchange [21]. The author wishes to thank Heng Guo and Peter Shor for valuable input on that question. The author also wishes to thank the moderators and creators of the stackexchange community, without which this paper not exist. In addition, the author would like to thank various members of the cs theory and mathematics community at UW-Madison for helpful conversations on related topics, especially Eric Bach, Jin-Yi Cai, Shuchi Chawla, Jordan Ellenberg, Tianyu Liu, Sebastien Roch, Dieter van Melkebeek and Stephen J. Wright. Encouraging correspondence with Heng Guo, Arne Riemers and Leen Stougie was also helpful. The author acknowledges the following grants: NSF RTG award DMS-1502553, NSF Award DMS-2023239 and the Prof. Amar G. Bose Research Grant.

References

[1] Scott Aaronson. $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$. In *Open problems in mathematics*, pages 1–122. Springer, 2016.

[2] Vicente Acuña, Paulo Vieira Milreu, Ludovic Cottret, Alberto Marchetti-Spaccamela, Leen Stougie, and Marie-France Sagot. Algorithms and complexity of enumerating minimal precursor sets in genome-wide metabolic networks. *Bioinformatics*, 28(19):2474–2483, 2012.

[3] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–12. ACM, 2019.

[4] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[5] David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Computational Geometry*, 8(3):295–313, 1992.

[6] Andrei Broder. Generating random spanning trees. In *30th Annual symposium on foundations of computer science*, pages 442–447. IEEE, 1989.

[7] Mary Cryan, Martin Dyer, Haiko Müller, and Leen Stougie. Random walks on the vertices of transportation polytopes with constant number of sources. *Random Structures & Algorithms*, 33(3):333–355, 2008.

[8] William H Cunningham and Jim Geelen. On integer programming and the branch-width of the constraint matrix. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 158–166. Springer, 2007.

[9] Rodney G Downey and Michael R Fellows. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

[10] Martin E Dyer. The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8(3):381–402, 1983.

[11] H-D Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical logic*. Springer Science & Business Media, 2013.

[12] Tomas Feder and Milena Mihail. Balanced matroids. In *Proceedings of the Twenty Fourth Annual ACM Symposium on the Theory of Computing*, pages 26–38. Citeseer, 1992.

[13] Komei Fukuda. Notes on matching polytope.

[14] Komei Fukuda, Thomas M Liebling, and François Margot. Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Computational Geometry*, 8(1):1–12, 1997.

[15] Julien Gagneur and Steffen Klamt. Computation of elementary modes: a unifying framework and the new binary approach. *BMC bioinformatics*, 5(1):1–21, 2004.

[16] Jim Geelen, Bert Gerards, and Geoff Whittle. Tangles, tree-decompositions and grids in matroids. *Journal of Combinatorial Theory, Series B*, 99(4):657–667, 2009.

[17] André Große, Jörg Rothe, and Gerd Wechsung. On computing the smallest four-coloring of planar graphs and non-self-reducible sets in p. *Information processing letters*, 99(6):215–221, 2006.

[18] Heng Guo and Mark Jerrum. Counting vertices of integer polytopes defined by facets. *arXiv preprint arXiv:2105.01469*, 2021.

[19] Pinar Heggernes. Treewidth, partial k-trees, and chordal graphs.

[20] Illya V Hicks and Nolan B McMurray Jr. The branchwidth of graphs and their cycle matroids. *Journal of Combinatorial Theory, Series B*, 97(5):681–692, 2007.

[21] Lorenzo Najt (https://cstheory.stackexchange.com/users/44995/lorenzo najt). Can one efficiently uniformly sample a neighbor of a vertex in the graph of a polytope? Theoretical Computer Science Stack Exchange.

[22] Mark Huber. Exact sampling from perfect matchings of dense nearly regular bipartite graphs. *arXiv preprint math/0310059*, 2003.

[23] Russell Impagliazzo and Avi Wigderson. P = BPP unless E has subexponential circuits: derandomizing the XOR lemma. In *Proceedings of the 29th STOC*, pages 220–229, 1997.

[24] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries, acm sympos. *Theory of computing*, 33:712–721, 2001.

[25] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, January 1986.

[26] Volker Kaibel. On the expansion of graphs of 0/1-polytopes. In *The Sharpest Cut: The Impact of Manfred Padberg and His Work*, pages 199–216. SIAM, 2004.

[27] Leonid Khachiyan. Transversal hypergraphs and families of polyhedral cones. In *Advances in Convex Analysis and Global Optimization*, pages 105–118. Springer, 2001.

[28] Leonid Khachiyan, Endre Boros, Konrad Borys, Vladimir Gurvich, and Khaled Elbassioni. Generating all vertices of a polyhedron is hard. In *Twentieth Anniversary Volume:*, pages 1–17. Springer, 2009.

[29] Samir Khuller and Vijay V Vazirani. Planar graph coloring is not self-reducible, assuming P≠ NP. *Theoretical Computer Science*, 88(1):183–189, 1991.

[30] Susan Margulies, Jing Ma, and Illya V Hicks. The Cunningham-Geelen method in practice: branch-decompositions and integer programming. *INFORMS Journal on Computing*, 25(4):599–610, 2013.

[31] Jiri Matousek and Bernd Gärtner. *Understanding and using linear programming*. Springer Science & Business Media, 2007.

[32] Frédéric Mazoit and Stéphan Thomassé. Branchwidth of graphic matroids. *Surveys in combinatorics*, 346(275-286):93, 2007.

[33] Peter McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17(2):179–184, 1970.

[34] Lorenzo Najt, Daryl DeFord, and Justin Solomon. Complexity and geometry of sampling connected graph partitions. *arXiv preprint arXiv:1908.08881*, 2019.

[35] Arne C Reimers and Leen Stougie. Polynomial time vertex enumeration of convex polytopes of bounded branch-width. *arXiv preprint arXiv:1404.5584*, 2014.

[36] Neil Robertson and Paul D Seymour. Graph minors. X. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.

[37] Robert Salomone, Radislav Vaisman, and Dirk P Kroese. Estimating the number of vertices in convex polytopes. In *Proceedings of the International Conference on Operations Research and Statistics*, pages 96–105. Citeseer, 2016.

[38] Claus-Peter Schnorr. On self-transformable combinatorial problems. In *Mathematical Programming at Oberwolfach*, pages 225–243. Springer, 1981.

[39] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[40] Stefan Schuster and Claus Hilgetag. On elementary flux modes in biochemical reaction systems at steady state. *Journal of Biological Systems*, 2(02):165–182, 1994.

A Another Application of the Hypercube Gadget

This section concerns the following kind of polytope:

Definition A.0.1. *A $\{0, \frac{1}{2}, 1\}$-polytope is a polytope $P(A, b) = \{Ax = b, x \geq 0\}$ such that the coefficients of each vertex in $P(A, b)$ are from the set $\{0, \frac{1}{2}, 1\}$.*

We can use the hypercube gadget to provide an alternative version of the proof of Theorem 3 from [18], which shows that sampling vertices of $\{0, \frac{1}{2}, 1\}$ polytopes is NP-hard. There are two main challenges they overcome in their proof: 1) identifying a polytope $P(G)$ associated to a graph G whose vertices correspond to feasible solutions for an NP-hard optimization problem on G, 2) showing that feasible solutions for that optimization problem are NP-hard to sample. As usual, the second step involves designing a gadget that modifies the original graph G so that uniform samples of feasible solutions of a related graph G' can be transformed into samples from feasible solutions on G that are optimal with high probability.

Instead of needing to design a problem specific gadget for the second step, which involves some creativity and careful bookkeeping, we can use the hypercube gadget. The key insight is that a function of the support of the vertices of $P(G)$ from step 1) coincides with the objective function for the optimization problem on G, connecting this set up to the optimization version of the RVP problem:

Optimization version of Restricted Vertex Problem (ORVP)
Input: An integer $m \times n$ matrix A, a vector $b \in \mathbb{Z}^m$ and $I \subseteq [n]$.
Output:$\max_{x \in \mathrm{Vert}(P(A,b))} \lvert \mathrm{supp}(x) \cap I \rvert$

We can summarize one key step of the Guo et al. proof as follows:

Theorem A.0.2. *(Following [18][Theorem 3]) ORVP is NP-hard for P $\{0, \frac{1}{2}, 1\}$ polytopes. More precisely, given a tuple (A, b, k, I) with the promise[2] that $P(A, b)$ is a $\{0, \frac{1}{2}, 1\}$ polytope, deciding whether $ORVP(A, b, I) \geq k$ is NP-hard.*

Proof. Consider the perfect 2-matching (P2M) polytope of a graph G, which is given by introducing a variable x_{uv} for each edge uv, and constraints:

$$0 \leq x_{uv} \leq 2$$

$$\sum_{uv \in E(G)} x_{uv} = 2, \forall u \in V$$

[2]Recall that the promise condition means that a decider only needs to produce the correct answer when the promise is true, and does not need to verify the promise.

To make this consistent with framework, we put this into slack form. Thus, the variables are $x_{uv}, y_{uv} \geq 0$ and they are subject to:

$$x_{uv} + y_{uv} = 2$$
$$\sum_{uv \in E(G)} x_{uv} = 2, \forall u \in V$$

Recall that an edge cover of a graph $G = (V, E)$ is a subset $S \subseteq E$ such that each vertex in V is incident to edge in S. By Proposition 2 of [18], the vertices of the P2M polytope of G, $P2M(G)$, correspond to edge covers of G consisting of a matching M, with $x_e = 2$ for $e \in M$, and vertex-disjoint odd cycles that are also disjoint from M, with $x_e = 1$ for edges in the odd cycles. Among such edge covers, there is a cover with $|V| = n$ edges iff there is an edge cover consisting only of disjoint odd-cycles. An odd cycle cover refers to such an edge cover that consists only of disjoint odd-cycles. We note that we can obtain the set underlying this combinatorial representation of a vertex by intersecting the support of the vertex with the x variables $\{x_{uv} : uv \in E\} = X$. If there are no odd cycle covers, then $|\text{supp}(x) \cap X| < n$ for all $x \in \text{Vert}(P2M(G))$, and if there is then there is a vertex x with $|\text{supp}(x) \cap X| = n$. Thus, the problem of determining if there is a vertex x of $P2M(G)$ with $|\text{supp}(x) \cap X| \geq n$ is equivalent to determining if there is an odd-cycle cover of G.

Guo et al show in [18][Theorem 3] that deciding if there is an odd-cycle cover on a bipartite graph is NP hard. Thus, the problem $\max_{x \in \text{Vert}(P2M(G))} |\text{supp}(x) \cap X|$ is NP-hard. Since this is a special case of ORVP problem, it follows that ORVP is NP hard on this class of polytopes. □

In order to apply our machinery to $\{0, 1/2, 1\}$ polytopes, we abstract theorem C.0.4 into the following theorem:

Lemma A.0.3. *Suppose $\mathcal{P}$ is a class of polytopes defined by linear inequalities, that is closed under hypercube gadget, and such that the optimization version of the restricted vertex problem is NP-hard on $\mathcal{P}$.*

Fix any α, $0 \leq \alpha < 1$. Suppose that there was a polynomial time probabilistic Turing machine M such that for each polytope $P(A, b)$ in $\mathcal{P}$, $M(A, b)$ outputs a sample from q_P, where q_P is any probability distribution on $\text{Vert}(P)$ with $||q_P - Uniform_{\text{Vert}(P)}|| \leq \alpha$. Then $\text{RP} = \text{NP}$.

Proof. Consider an instance $(((A, b), I), k)$ of ORVP with $P(A, b) \in \mathcal{P}$ a polytope in $\mathbb{R}^n$.

Suppose that $\max_{x \in \text{Vert}(P)} |\text{supp}(x) \cap I| = m$. We let $S = \text{argmax}_{x \in \text{Vert}(P)} |\text{supp}(x) \cap I|$.

Construct $P^{d,I} =: P^d$; then, by lemma 5.0.2, P^d has 2^{md} vertices above any vertex of P in S. Above any other vertex, it has at most $2^{(m-1)d}$ vertices.

Let $H = \pi^{-1}(S) \cap \text{Vert}(P^d)$ and $N = \text{Vert}(P^d) \backslash H$. Since $P(A, b)$ has at most 2^n vertices, $|N| \leq 2^n 2^{(m-1)d}$ and $|H| \geq 2^{md}$. Therefore, we have $|H| \geq D|N|$ with $D = 2^{d-n}$, so $\frac{|H|}{|H|+|N|} \geq \frac{D}{1+D}$, and so a uniform sample from the vertices of P^d projects under π to an element of S with probability at least $\frac{2^{d-n}}{1+2^{d-n}} \geq 1 - 2^{-m}$ for $d = n + m$. If we take m so that $1 - 2^{-m} \geq 1 - \alpha/2$, then a sample from M applied to P^d has probability at least $\alpha/2$ of sampling from S. By amplification, we can raise this to $3/4$ probability, and we obtain a randomized algorithm for the instance of ORVP which gives the correct answer with probability $3/4$. Since only false negatives are possible in this algorithm, since we always obtain a candidate witness in the YES case, this implies that $\text{RP} = \text{NP}$. □

We can now use the hypercube gadget to complete the proof of Theorem 3 of Guo et al.:

Theorem A.0.4. *The uniform sampling problem is NP hard on polytopes promised to be $\{0, \frac{1}{2}, 1\}$-polytopes.*

Proof. Let $\mathcal{H}$ denote the class of $\{0, \frac{1}{2}, 1\}$ polytopes. First, we show that that the hypercube gadget preserves $\mathcal{H}$. Consider $P \in \mathcal{H}$. By lemma 5.0.2, $\text{Vert}(P^{I,d}) = \bigcup_{x \in \text{Vert}(P)} \bigcup_{y \in \text{Vert}(H_x)} \{(x, y)\}$ for $x \in \text{Vert}(P)$, $x \in \{0, \frac{1}{2}, 1\}$, and $H_x = \{(y, y^s) : y, y^s \geq 0, y_{i,j} + y^s_{i,j} = x_i\}$. To conclude, we note that H_x is a $\{0, \frac{1}{2}, 1\}$ polytope for any $x \in \{0, \frac{1}{2}, 1\}^n$. Since, theorem A.0.2 shows RVP is NP hard on $\mathcal{H}$, it follows by lemma A.0.3 that the uniform sampling problem is NP hard. □

Remark A.0.5. *Guo et al. are motivated by the problem of vertex sampling a 0/1 polytope, meaning a polytope such that each coordinate of each vertex is in* $\{0,1\}$*. Our approach does not work in that setting, since ORVP is solvable by maximizing* $\sum_{i\in I} x_i$ *via linear programming.*

Remark A.0.6. *The two matching polytope is given by introducing a variable* x_{uv} *for each edge* uv*, and constraints:*

$$0 \leq x_{uv} \leq 2$$
$$\sum_{uv\in E(G)} x_{uv} = 2, \forall u \in V$$

The second constraint has constraint matrix given by the incidence matrix of underlying directed graph. Thus, using the calculations in theorem F.2, the branchwidth of this linear system is at least the branchwidth of the underlying undirected graph. Since, by considerations similar to appendix F, Courcelle's theorem shows that the odd cycle cover problem is in P on directed graphs whose underlying graph has bounded branchwidth, it follows that this family of polytopes do not provide an alternative proof that vertex sampling is hard when the branchwidth is bounded.

B Non Self-Reducibility

In this section we examine the self-reducibility of a natural p-relation associated to the set of vertices of polytopes. The argument is similar to the one on [29]; in particular the sense of self-reducibility that we use in definition B.0.3 is in the same as in [25, 29, 38]. In addition, Theorem 3.5 of [17] provides some general conditions under which one can show that a set is not self-reducible, unless P $\neq$ NP, showing that there is an abundance of examples of such p-relations. For us, the non self-reducibility of this p-relation only matters in as much as it means that we cannot directly appeal to the equivalence of counting and sampling to prove hardness of counting vertices in appendix C.

We fix some finite alphabet Σ, and for $w \in \Sigma^*$, where Σ^* denotes all finite length words that can be spelled with characters from Σ, including the empty word. we let $|w|$ denote the number of symbols in w. Recall that the notion of a p-relation is meant to encode the idea of a (problem, solution) pair, where solutions are not too big and a proposed solution to a problem can be efficiently verified as a solution or not. That is:

Definition B.0.1 (p-relation [25])**.** *A relation* $R \subseteq \Sigma^* \times \Sigma^*$ *is a p-relation if :*

1. *There is a polynomial* $p(n)$ *such that* $(x, y) \in R$ *implies* $|y| \leq p(|x|)$*.*
2. *There is a polynomial time Turing machine that decides* R*, i.e.* $M(x\#y) = 1$ *if* $(x, y) \in R$ *and* $M(x\#y) = 0$ *if* $(x, y) \notin R$*. Here* $\#$ *is some character not in* Σ*.*

For $x \in \Sigma^*$*, we define* $R(x) := \{y \in \Sigma^* : (x, y) \in R\}$*.*

As an example of a p-relation is given by taking the x-coordinate to encode a graph G, and the y-coordinate to encode a perfect matching of G. If G has n edges, with some canonical order coming from the encoding, y could be represented by a binary string indicating which edges are in the graph. The p-relation for the vertices of a polytope, namely (Polytope in equational form, vertex), will be obtained by encoding a vertex of the polyhedron $P(A, b) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ by the support vector as a string in $\{0, 1\}^n$. We state this precisely below in lemma B.0.2. The previously mentioned papers on the vertex enumeration problem, namely [13, 35], also use this encoding.

Lemma B.0.2. *We use* $\Sigma = \{0, 1, ;, (,)\}$ *and assume some standard encoding of integers, vectors and matrices using those symbols. The relation defined by*

$$R_{Polytope} = \{((A, b), s) : A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^m, \exists v \in \mathrm{Vert}(P(A, b)), s = \mathrm{supp}(v) \in \{0, 1\}^n, P(A, b) \text{ a polytope}\}$$

is a p-relation.

Proof. First, we check that R is a p-relation. Since s is a binary vector of length n, clearly $|s| \leq |(A,b)|$. Thus, it suffices to show that $((A,b),s) \in R$ can be checked in time polynomial in $|(A,b)|$, which requires checking whether there is a vertex of $P(A,b)$ with support exactly s. Note that this is *not* the restricted vertex problem, which asks for the support vector to be larger than s.

Consider any vector $s \in \{0,1\}^n$. Let $S = \{i : s_i > 0\}$. Let A_s denote the set of columns of A of indices i with $s_i \neq 0$. Suppose that there is a solution $A_s x_s = b$ with $x_s \geq 0$ and the columns of A_s are linearly independent. Define the point x to agree with x_s on the coordinates in S, and to be zero elsewhere. Then, x is a feasible point and, by lemma 4.0.7, it is also a vertex.

On the other hand, if A_s is not linearly independent, then by lemma 4.0.7 there is no vertex witih support S. If A_s is linearly independent, but the solution to $A_s x_s = b$ has some negative coordinates, or there is no solution at all, then there are no feasible points with support s.

Therefore, there is a vertex with support vector s iff A_s is linearly independent and there is a solution $x_s \geq 0$ to $A_s x_s = b$. Since we can efficiently check linear independence of the columns of A_s using Gaussian elimination and the existence of a solution to $A_s x_s = b$ with $x_s \geq 0$ by using any polynomial time linear programming algorithm, we can check this condition in polynomial time. □

We now turn to self-reducibility (definition B.0.3). Roughly speaking, a p-relation is self-reducible if the set of solutions to an instance x can be partitioned using the solutions to a small set of subproblems of the same kind. One of the reasons to be interested in self-reducible p-relations is that, for self-reducible relations, the approximate sampling and approximate counting problems are equivalent [25]. For instance, the graph matching p-relation is self-reducible, since given an edge e of G, the matchings of G can be partitioned into the matchings containing e and those not containing e.

Definition B.0.3 (Self-reducibility of a p-relation [25])**.** *A p-relation $R \subseteq \Sigma^* \times \Sigma^*$ is said to be self reducible if there are polynomial time computable functions $l : \Sigma^* \to \mathbb{N}$, $\sigma : \Sigma^* \to \mathbb{N}$ and $\psi : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that:*

1. $\sigma(x) = O(\log(|x|))$
2. $y \in R(x)$ *implies* $|y| = l(x)$
3. $l(x) > 0$ *implies* $\sigma(x) > 0$.
4. $|\psi(x,w)| \leq |x|$.
5. $\forall x \in \Sigma^*$, $R(x) = \bigcup_{w \in \{0,1\}^{\sigma(x)}} \{wy : y \in R(\psi(x,w))\}$
6. $l(\psi(x,w)) = \max\{0, l(x) - |w|\}$.

The last two conditions can equivalently be expressed as: for all $x \in \Sigma^*, y = y_1, \ldots y_n \in \Sigma^*$, $(x,y) \in R$) iff $(\psi(x, y_1 \ldots y_{\sigma(x)}), y_{\sigma(x)+1} \ldots y_n) \in R$.

One strategy for showing that self-reducibility of some particular p-relation implies $NP = P$ relies on the following search to decision reduction:

Lemma B.0.4 (Implicit in [29])**.** *Suppose that R is a self-reducible p-relation, and suppose that there is a polynomial time algorithm for determining if $R(x) = \{y : (x,y) \in R\}$ is non-empty. Then, there is a polynomial time algorithm that, given x and $z \in \{0,1\}^k$, decides if there is a $y \in R(x)$ such that $y = zz'$ for some $z' \in \{0,1\}^{l(x)-k}$.*

.

Proof. The algorithm, in pythonic pseudocode, follows. If z is a string, then $z[:j]$ denote the first j characters, and $z[j+1:]$ denotes the remaining characters.

```
1 begin{lstlisting}
2 f is_empty(set):
3   return len(set) == 0
4
5 f has_solution(x, z):
6   if z == '':
7       return not is_empty(R(x))
```

```
if sigma(x) <= |z|:
        return has_solution(R(psi(x, z[:sigma(x)] , z[sigma(x) + 1:])))
else:
    found_solution = False
    for w in {0,1}^{sigma(x) - |z|}:
        found_solution +=  not is_empty(R(psi(x, z[:sigma(x)] + w)))
    return found_solution
```

We note that because $\sigma(x) = O(log(x))$ the for-loop in the else branch takes polynomial time. In addition, because $|z| \leq l(x)$, and the length decreases on each recursive call, there are at most $l(x)$ recursive calls. Thus, the entire algorithm runs in polynomial time. □

We will next show that this search to decision reduction can be used to prove the existence of a solution with a prescribed set of 1, i.e. whose support, if interpreted as a function on $[l(x)]$, contains a specific set. Ultimately, this will be connected to the restricted vertex problem.

Lemma B.0.5. *Suppose that R is a self-reducible p-relation, and suppose that there is a polynomial time algorithm for determining if $R(x) = \{y : (x, y) \in R\}$ is non-empty. Furthermore, suppose that R is such that, for instance x and permutation σ, there is another instance x_σ such that $R(x_\sigma) = \{\sigma(y) : y \in R(x)\}$, such that x_σ can be calculated in polynomial time from (x, σ) .[3] Then there is a polynomial time algorithm that, given x and $z \in \{0, 1\}^{l(x)}$, determines if there is a $y \in R(x)$ with $y \geq z$.*

Proof. Let 1^j denotes the all ones string of length j, and 0^j denote the all zeros string of length j. Given z, we can find a permutation σ so that $\sigma(z) = 1^j 0^{l(x)-j}$. There is then an element $y \in R(x_\sigma)$ so that $y \geq 1^j 0^{l(x)-j}$ iff there is an element $y' \in R(x)$ so that $y \geq z$; this is because $y \in R(x_\sigma)$ iff $y = \sigma(y')$ for $y \in R(x)$, and $\sigma(y') \geq 1^j 0^{l-j}$ iff $y' \geq \sigma^{-1}(1^j 0^{l-j}) = z$. Finally, since $y \geq 1^j 0^{l(x)-j}$ is equivalent to $y[: j] = 1^j$, using lemma B.0.4, we can decide the existence of such a y in polynomial time.

□

To use this in our setting, we need the following lemma:

Lemma B.0.6. *Given a polyhedron $P(A, b)$, there is a polynomial time algorithm to check whether $P(A, b)$ is a polytope, and that it has a vertex.*

Proof. Checking that it is a polytope amounts to calculating $\max x_i$ and $\min x_i$ for all i, which can be done using a polynomial time linear program. A polytope has a vertex iff it is non-empty, and we can again use linear programming to check that there is a feasible solution. □

Theorem B.0.7. *$R_{Polytope}$ is not self-reducible unless* P = NP.

Proof. By lemma B.0.6, we have a polynomial time algorithm to check whether $R_P((A, b)) = \emptyset$. In addition, we note that for any $x = (A, b) \in \Sigma^*$ and $\sigma \in S_{l(x)}$, the instance (x_σ) is given by (AS_σ, bS_σ), where S_σ is the permutation matrix of σ. Thus, if R was self-reducible, all of the hypothesis of appendix B would be satisfied. However, this implies that the restricted vertex problem would be solvable in polynomial time, which would imply P = NP.

□

For self-reducible structures, there is a well-known equivalence between counting and sampling [25]. Despite the non-self reducibility, we show that the hypercube gadget is enough to prove that the marginal counts necessary for sampling could be made using a Turing machine that can count vertices, see lemma C.0.3.

Remark B.0.8. *This is similar to the situation in [34] with connected partitions and simple cycles: there the structures are not self reducible, but one can use a chain of bigons gadget to relate counting and marginal counts. We remark that there is an analogy between the hypercube gadget and the chain of bigons gadgets used that investigation: indeed, as mentioned in §6.2 the hypercube gadget can be thought of as a series of parallel extensions followed by series extensions, which, in the graph context, is the same as the chain of bigons gadget.*

[3] This awkward sounding condition just amounts to settings where we can order the variables however we like, for instance by relabelling the edges of a graph, or reordering the columns of the constraint matrix of a linear program.

C Intractability of Approximate Counting

In this section we show that, under the standard complexity theory assumption that RP $\neq P$ (see [1, 23]), there do not exist approximation algorithms for counting vertices of a polytope given in equational form. As a byproduct we regain an equivalence between counting and sampling, despite the non self reducibility of the p-relation. We again use the hypercube gadget to reduce to RVP. We use the following notation to simplify the statement of our theorem:

Definition C.0.1 (Randomized polynomial-time constant-factor approximation algorithm (r-RPCA))**.** *An r-RPCA for a function problem $T : \Sigma^* \to \mathbb{N}$ is a probabilistic Turing machine A with the following property: A runs in time poly$(|x|)$ and has $\mathbb{P}(r^{-1}T(x) \leq A(x) \leq rT(x)) \geq 2/3$.*

Lemma C.0.2. *Let A be any matrix. If A' is obtained from A by adding a row to A consisting of a vector that is nonzero in exactly one entry, then* $\mathrm{bw}(A') \leq \max(2, \mathrm{bw}(A))$.

Proof. A' is obtained from A by a series extension, and then by deleting the added column. Thus the lemma follows from lemma F.2.4 and lemma 6.1.1. □

The following lemma shows that we can turn approximate vertex counters into approximate counters for marginal counts, without losing too much in the approximation ratio.

Lemma C.0.3. *For any (A, b) and $I, J \subseteq [n]$, define* $\mathrm{Vert}(P(A,b))_{I,J} := \{v \in \mathrm{Vert}(P(A,b)), \mathrm{supp}(v) \cap I = I, \mathrm{supp}(v) \cap J = \emptyset\}$. *Suppose that there is an r-RPCA of any polytope given in the form $P(A, b)$ that achieves approximation ratio of $r > 0$ with probability $\geq 2/3$. Then, for any $\epsilon > 0$, there is an re^{ϵ}-RPCA for the counting problem $((A, b), I, J) \to |\mathrm{Vert}(P(A,b))_{I,J}|$.*

Proof. We define $P_{I,J,d}$ by $P_{I,J,d} := \{x \in P_{I,J,d} : x_j = 0, \forall j \in J\}$; this is equivalent to defining $Q_J = \{x \in P : x_j = 0, \forall j \in J\}$ and then $P_{I,J,d} = (Q_J)_{I,d}$. We have $\mathrm{bw}(Q_J) \leq \max(\mathrm{bw}(P), 2)$ by lemma C.0.2, thus $\mathrm{bw}(P_{I,J,d}) \leq \max(\mathrm{bw}(Q_J), 2) \leq \max(\mathrm{bw}(P), 2$ by lemma 6.2.1. Let $a := |\mathrm{Vert}(P_{I,J,d})|$ and $a_k := |\{v \in \mathrm{Vert}(P) : |\mathrm{supp}(v) \cap I| = k, \mathrm{supp}(v) \cap J = \emptyset\}|$. Let $s = |I|$. From corollary 5.0.3 we have that $a = \sum_{k=0}^{s} 2^{dk} a_k$. Suppose that $\hat{a}$ is within multiplicative error r of a. We will argue that $\hat{a}/2^{ds}$ is within multiplicative error re^{ϵ} of a_s, provided that $d = n + \lceil \log_2(1/\epsilon) \rceil$ and that $a_s \neq 0$. The case $a_s = 0$ can be checked in BPP given the RPCA for counting vertices, by considering the rate of growth of a as a function of d.[4] Thus, we can assume the algorithm has already determined if $a_s = 0$ and terminated with its estimate for a_s as 0 if so.

We now check the assertion on multiplicative error of the estimator $\hat{a}/2^{sd}$. We are given that $a/r \leq \hat{a} \leq ra$, into which we can substitute $a = \sum_{k=0}^{s} 2^{dk} a_k = 2^{sd}(a_s + \sum_{k=0}^{s-1} 2^{d(k-s)})$ and then divide by 2^{sd}, producing: $\frac{1}{r}(a_s + \sum_{k=0}^{s-1} a_k 2^{(k-s)d}) \leq \hat{a}/2^{sd} \leq r(a_s + \sum_{k=0}^{s-1} a_k 2^{(k-s)d})$. Finally, we use that $\sum_{k=0}^{s} a_k \leq |\mathrm{Vert}(P)| \leq 2^n$, $a_s \geq 1$ nd $2^{(k-s)d} \leq 2^{-d}$ for $k = 0, \ldots, s-1$ to obtain the upper bound in the following, whereas the lower bound is trivial since $e^{\epsilon} > 1$:

$$\frac{1}{re^{\epsilon}} a_s \leq \frac{1}{r} a_s \leq \frac{1}{r}(1 + \sum_{k=0}^{s-1} \frac{a_k}{a_s} 2^{(k-s)d}) a_s \leq \hat{a}/2^{sd} \leq r(1 + \sum_{k=0}^{s-1} \frac{a_k}{a_s} 2^{(k-s)d}) a_s \leq r(1 + 2^n 2^{-d}) a_s \leq r(1+\epsilon) a_s \leq re^{\epsilon} a_s$$

Since the overhead in this construction is polynomial, in that we only need to write down the Poly$(|A|)$ equations defining $P_{I,J,d}$ and then do some arithmetic with integers that have polynomially many bits, the entire algorithm takes polynomial time in Poly$(|A|)$. □ □

We now prove the main theorem of this section.

Theorem C.0.4. *Unless* NP = RP*, for all $r > 0$ there is no r-RPCA for counting vertices of a polytope $P(A, b)$, even if A has branchwidth ≤ 4.*

[4] Take $d = n + 1 + \log_2(r)$. If $a_s = 0$, then $a = \sum_{k=0}^{s-1} 2^{dk} a_k \leq 2^n 2^{d(s-1)} = 2^{n+(n+1+\log_2(r))(s-1)} = 2^{ns+s-1} r^{s-1}$. If $a_s \geq 1$, then $a \geq a_s 2^{ds} \geq 2^{(n+1+\log_2(r))s} = 2^{ns+s} 2^r$. Thus, we have a promised multiplicative gap of $2r$, so an approximation algorithm with ratio r that succeeded 2/3 of the time would put deciding if $a_s = 0$ into BPP.

Proof. Suppose that there was an r-RPCA. Then by lemma C.0.3 there would be a $2r$-RPCA for the marginal counts, $|\mathrm{Vert}(P(A,b))_{I,J}|$. Given an instance $((A,b),I,J)$ of RVP, a $2r$-RPCA could determine whether $|\mathrm{Vert}(P(A,b))_{I,J}| \geq 1$ or $|\mathrm{Vert}(P(A,b))_{I,J}| = 0$ in BPP, solving the RVP problem, which we know to be NP-hard. This would put NP $\subseteq$ BPP, from which it is known that NP = RP follows. □

D Additional Branchwidth Lemmas

These lemmas will be used in other parts of the appendix.

Lemma D.0.1. *Let A be any matrix, and let A' be obtained from A by adding a new column. That is, $A' = \begin{pmatrix} A & v \end{pmatrix}$. Then* $\mathrm{bw}(A) \leq \mathrm{bw}(A') \leq \max(\mathrm{bw}(A)+1, 2)$.

Proof. The first inequality, $\mathrm{bw}(A) \leq \mathrm{bw}(A')$, was proven already in lemma 6.1.6. For the second inequality, suppose the columns of A are indexed by $[n]$. Let (T,ν) be an minimal width branch decomposition for A, and split it at the leaf n to obtain a branch decomposition for A', (T',ν'); let λ,λ' be the respective connectivity functions. Let $e \in E(T)$. Then e defines a partition of columns of A, $\{L,R\}$, and also a partition of A', say $\{L',R'\}$. Without loss, we can assume that the $n+1st$ column, v, is in R'. Then $\mathrm{Span}(L') \cap \mathrm{Span}(R') = \mathrm{Span}(L) \cap \mathrm{Span}(R,v) \subseteq (\mathrm{Span}(L) \cap \mathrm{Span}(R)) + \mathrm{Span}(v)$, and so the dimension of the intersection of the spans increases by at most one. In terms of the connectivity function, we have proven that $\lambda'(e) \leq \lambda(e)+1$ for $e \in E(T)$. For edges $e \in E(T') \setminus E(T)$, one of the blocks has size one, so $\lambda'(e) \leq 2$. This proves the second inequality. □

Lemma D.0.2. *Let $A \in \mathbb{R}^{n\times m}$, and define $A' = \left(\begin{smallmatrix} A \\ v \end{smallmatrix}\right)$ for any vector v; that is, A' is obtained from A by adding an additional row. Then* $\mathrm{bw}(A) - 1 \leq \mathrm{bw}(A') \leq \mathrm{bw}(A)+1$.

Proof. Let $[n]$ index the columns of A. Let (T,ν) be an optimal branch decomposition for A, and consider it as a branch decomposition for A'; let λ,λ' be the respective connectivity functions. Then, for any partition (L,R) of $[n]$, we have $\mathrm{Span}(A'_L) \cap \mathrm{Span}(A'_R) \subseteq \mathrm{Span}(i(\mathrm{Span}(A_L,A_R), e_{m+1})$, where $i : \mathbb{R}^m \to \mathbb{R}^{m+1}$ extends a vector by zero: $i(v) = (v,0)$. Thus, we have that $\lambda'(e) \leq \lambda(e)+1$ for all $e \in E(T)$, from which $\mathrm{bw}(A') \leq \mathrm{bw}(A)+1$ follows. To obtain the lower bound, we observe that for any matrices M and $\left(\begin{smallmatrix} M \\ v \end{smallmatrix}\right)$ $\mathrm{rk}(M) \leq \mathrm{rk}(M') \leq \mathrm{rk}(M)+1$, hence for any $e \in E(T)$ inducing a partition (L,R) of the columns, we have $\lambda'(e) = \mathrm{rk}(A'_L) + \mathrm{rk}(A'_R) - \mathrm{rk}(A') + 1 \geq \mathrm{rk}(A_L) + \mathrm{rk}(A_R) - \mathrm{rk}(A) - 1 + 1 = \lambda(e) - 1$. □

Example D.0.3. *Both situations in lemma D.0.2 can happen; that is, the inequality is tight:*

1. $\mathrm{bw}(\left(\begin{smallmatrix} 1&1&0 \\ 1&1&0 \\ 0&1&0 \end{smallmatrix}\right)) = 1$, *while* $\mathrm{bw}(\left(\begin{smallmatrix} 1&1&0 \\ 1&1&0 \end{smallmatrix}\right)) = 2$.
2. $\mathrm{bw}(\left(\begin{smallmatrix} 1&0&0 \\ 0&0&0 \end{smallmatrix}\right)) = 1$, *while* $\mathrm{bw}(\left(\begin{smallmatrix} 1&0&0 \\ 0&0&0 \\ 0&1&1 \end{smallmatrix}\right)) = 2$.

Proof. There is a unique cubic tree with three leaves. The partitions correspond to each of the three partitions of $[n]$ into a 2-partition with non-empty blocks. Therefore, the branchwidth is $\max_{i\in[3]} \dim(\mathrm{Span}(A_i) \cap \mathrm{Span}(A_{[3]\setminus i})) + 1$. This can be computed for each of the four matrices. □

Lemma D.0.4 (Row Space Lemma)**.** *The A be any matrix, and let a be a vector in the row space of A. The, let A' be the matrix $\left(\begin{smallmatrix} A \\ a \end{smallmatrix}\right)$; that is, a matrix obtained by adding adding a as a row to A'. Then* $\mathrm{bw}(A') = \mathrm{bw}(A)$.

Proof. We note that if $M'$is obtained from M by adding a row from the row space, then $\mathrm{rk}(M') = \mathrm{rk}(M)$. This implies a $[n]$-labelled cubic tree has the same width for both A and A'. □

The following is also useful:

Lemma D.0.5 (Rank Bound)**.** *Let A be any matrix. Then* $\mathrm{bw}(A) \leq \mathrm{rk}(A)+1$, *where* $\mathrm{rk}(A)$ *is the rank of the matrix.*

Proof. Any branch decomposition will prove this, since $\dim(\mathrm{Span}(A_{L_e}) \cap \mathrm{Span}(A_{R_e})) + 1 \leq \dim(\mathrm{Span}(A)) + 1$. □

Lemma D.0.6. *If U is any invertible matrix, then* $\mathrm{bw}(UA) = \mathrm{bw}(A)$.

Proof. This follows because $\mathrm{Span}(UA_I) \cap \mathrm{Span}(UA_{I^c}) = U(\mathrm{Span}(A_I) \cap \mathrm{Span}(A_{I^c}))$, for any subset $I \subseteq [n]$, where U acts on subspaces in the usual way. □

The next lemma allows us to remove constant variables from a system of equations; that is, variables x_i so that $Ax = b$ imposes the constraint $x_i = b_j$ for some j. We will use this later on in order to bound the branchwidth of the system of linear equations that are used for the hardness of the restricted vertex problem.

Lemma D.0.7 (Removing constant variables)**.** *Let A be any matrix. Suppose that there is a row of A containing a single non-zero entry in the ith spot. Then if A' is A with the ith column deleted,* $\mathrm{bw}(A') = \mathrm{bw}(A)$.

Proof. Without loss, we can assume that i is the last column, n, and that the row is the last row, say m. Using lemma D.0.6 we can clear the entire nth column, except for the entry in the mth row, without changing the branchwidth. Thus, we can assume that A is of the form: $\left(\begin{smallmatrix} B & 0 \\ 0 & 1 \end{smallmatrix}\right)$. We have that $A' = \left(\begin{smallmatrix} B \\ 0 \end{smallmatrix}\right)$, thus $\mathrm{bw}(A') = \mathrm{bw}(B)$ by lemma D.0.4.

It remains to show that $\mathrm{bw}(B) = \mathrm{bw}(A)$. We know from lemma D.0.4 and lemma 6.1.1 than $\mathrm{bw}(A) \geq \mathrm{bw}(B)$. To establish that $\mathrm{bw}(B) \geq \mathrm{bw}(A)$, consider any branch decomposition (T, ν) of A. We obtain a branch decomposition of B by merging the leaf labelled n with its sibling and parent, and labelling the result $n-1$. Call the resulting branch decomposition of B $\bar{T}$. For any $e \in E(\bar{T})$, it induces a partition L and R of $[n-1]$, and assuming that $n-1 \in R$, considering e an edge of T, it induces the partition L, $R \cup \{n\}$. Then, if $i : \mathbb{R}^{n-1} \to \mathbb{R}^n$ is the natural embedding into the first $n-1$ coordinates, then we have $i(\mathrm{Span}(B_L)) = \mathrm{Span}(A_L)$ and $i(\mathrm{Span}(B_R)) + e_n = \mathrm{Span}(A_R)$. In particular, we have that $i(\mathrm{Span}(B_L) \cap \mathrm{Span}(B_R)) \subseteq \mathrm{Span}(A_L) \cap \mathrm{Span}(A_R)$. Since $\dim(\mathrm{Span}(A_L) \cap \mathrm{Span}(A_R)) \leq \mathrm{bw}(A) - 1$ by definition, the claim is proven. □

E Restricted Vertex Problem Under Branchwidth Constraints

This section reviews some details of Theorem 10 from [35], which appeared above as Theorem 7.0.2.

We require the following auxiliary problem:

Single Constraint Integer Programming Feasibility (SCIPF)
Input: A non-negative vector $a \in \mathbb{Z}^n$ and an integer b.
Output: Whether there is an $x \in \{0,1\}^n$ satisfying $a \cdot x = b$

SCIPF is a reformulation of the subset sum problem, and is thus weakly NP-complete [35]. Reimers et. al. [35] encode SCIPF into a polyhedron P with a designated variable u, such that P has a vertex with $u > 0$ iff there is a solution $ax = b$, with $x \in \{0,1\}^n$. (The authors of [35] use x_0 instead of u, I have opted for the latter here to clarify notation.) A definition of this polytope follows, after which we verify its branchwidth and the claimed properties.

Definition E.1. *Given an instance of SCIPF, $ax = b$, define $S(a,b)$ to have non-negative coordinates $(x, \bar{x}, y, \bar{y}, u, z)$ of dimensions $(n, n, n, n, 1, 1)$ and to be constrainted by the following equations:*

$$a \cdot x + u - (b+1)z = 0 \tag{E.1}$$

$$\mathbb{1} \cdot y + \mathbb{1} \cdot x - nz = 0 \tag{E.2}$$

$$x_i + \bar{x}_i = z, i \in \{1, 2, \ldots, n\} \tag{E.3}$$

$$y_i + \bar{y}_i = z, i \in \{1, 2, \ldots, n\} \tag{E.4}$$

$$2z - u = 1 \tag{E.5}$$

The $\bar{x}, \bar{y}$ are slack variables and $\mathbb{1}$ denotes the all ones vector.

We will now bound the branchwidth of $S(a,b)$:

Proposition E.0.1. *The branchwidth of $P = S(a,b)$ is ≤ 4.*

Proof. We let A be the coefficient matrix of P with the column z deleted. We will show that A has branch width ≤ 3. The proposition then follows from lemma D.0.1. After deleting the z variable, the equation eq. (E.5) becomes $u = 1$. We then appeal to lemma D.0.7 to remove u from the system without changing the branchwidth any further. After making these deletions, the system $A(x, \bar{x}, y, \bar{y}) = 0$ is described by:

$$\begin{aligned} a \cdot x &= 0 \\ \mathbb{1} \cdot (y + x) &= 0 \\ x_i + \bar{x_i} &= 0, i \in \{1, 2, \ldots, n\} \\ y_i + \bar{y_i^s} &= 0, i \in \{1, 2, \ldots, n\} \end{aligned}$$

In particular, ordering the variables as $(x, \bar{x}, y, \bar{y})$ the matrix A is given by the following, where the variables are listed in order $x_1, \ldots, x_n, \bar{x_1}, \ldots, \bar{x_n}, y_1, \ldots, y_n, \bar{y_1}, \ldots, \bar{y_n}$):

$$A = \begin{pmatrix} a_1 & \ldots & a_n & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 1 & \ldots & 1 & 0 & \ldots & 0 & 1 & \ldots & 1 & 0 & \ldots & 0 \\ 1 & \ldots & 0 & -1 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & 1\ldots & 0 & 0 & -1\ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ & & & & & \vdots & & & & & & \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 1 & \ldots & 0 & -1 & \ldots & 0 \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & 1\ldots & 0 & 0 & -1\ldots & 0 \\ & & & & & \vdots & & & & & & \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{pmatrix}$$

The matrix A is obtained from the matrix

$$A' = \begin{pmatrix} a_1 & \ldots & a_n & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 1 & \ldots & 1 & 0 & \ldots & 0 & 1 & \ldots & 1 & 0 & \ldots & 0 \end{pmatrix}$$

by repeated series extensions, and a permutation of the columns. Therefore, lemma 6.1.6 and lemma 6.1.2 show that $\mathrm{bw}(A) \leq \max(\mathrm{bw}(A'), 2)$. Finally, we observe that that $\mathrm{bw}(A') \leq 3$ by lemma D.0.5.

□

Reimers et al related a SCIPF instance (a, b) to the existence of certain vertices of $S(a, b)$.

Theorem E.1 ([35] Theorem 8)**.** *The existence of a solution to $\{x \in \{0,1\}^n : ax = b\}$ is equivalent to $|\{(x, y, u, z) \in \mathrm{Vert}(S(a,b)) : u > 0\}| \neq 0$.*

Proof. See [35]. □

So far $S(a, b)$ is a potentially unbounded polyhedron. To obtain a polytope, we need the notion of the size of a linear system, briefly reviewed in the following subsection.

E.1 The size of a linear system

Definition E.2 (Size)**.** *The size of a rational number p/q, stored as a pair of integers p and q, is* $\mathrm{size}(p/q) = 1 + \lceil \log_2(p) + 1 \rceil + \lceil \log_2(q) + 1 \rceil$*. The size of a matrix $A = (\alpha_{i,j}) \in M_{m,n}(\mathbb{Q})$ is defined by $mn + \sum_{i,j} \mathrm{size}(\alpha_{ij})$.*

Lemma E.2 (Vertices of polytopes are polynomial in size)**.** *Consider a polytope* $P(A,b) = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ *where* A *has* m *rows. If* $v \in \text{Vert}(P(A,b))$*, then for any* $i \in [n]$*,* $\text{size}(v_i) \leq m(n^2(2+4\text{size}(A)) + \text{size}(b))$*. In particular,* $|v_i| \leq 2^{m(n^2(2+4\text{size}(A))+\text{size}(b))}$ *and, if* $v_i > 0$*, then* $|v_i| \geq 2^{-m(n^2(2+4\text{size}(A))+\text{size}(b))}$*.*

Proof. Let B be a basis corresponding to the vertex v, so A_B denotes the corresponding minor of A and we have $v_B = A_B^{-1}b$ with $v_i = 0$ for $i \notin B$. By proposition 3.2 in [39], we have that $\text{size}(\det(A_B)) \leq 2\text{size}(A_B) \leq 2\text{size}(A)$. In particular, if M is any minor of A_B, we have $\text{size}(\det(M)) \leq 2\text{size}(A)$. Since each entry in A_B^{-1} is of the form $\det(M)/\det(A_B)$, for some minor M of A, we have that the size of each entry of A_B^{-1} is bounded by $1 + 4\text{size}(A)$. Thus, $\text{size}(A_B^{-1}) \leq n^2 + n^2(1 + 4\text{size}(A))$. It follows that the size of any entry of $A_B^{-1}b$ is $\leq (n^2(2 + 4\text{size}(A)) + \text{size}(b))$ which is the main claim of the lemma. The other two claims follow. □

Recall that our goal is to replace the potentially unbounded SCIPF polyhedron with a polytope. We will do this by new constraints that do not cut out any of the original vertices, and which do not increase the branchwidth. To wrap up the proof, in the next section we'll use the support condition in the restricted vertex problem to focus only on vertices that were not introduced by the new constraint.

Lemma E.3. *Let* $P(A,b)$ *be a polyhedron in* $\mathbb{R}^n$*. Take* Γ *to be any real number such that for all vertices of* P*, each coordinate of* v *is* $< \Gamma$*. For instance, by lemma E.2,* $\Gamma = 2^{n^2(1+2size(A))size(p)} + 1$ *suffices. Let* $Q = \{(x,w) \in \mathbb{R}^{n+n}_{\geq 0} : Ax = b, x_i + w_i = \Gamma \text{ for } i = 1,2,3\ldots,n\}$*. Then,* Q *is a polytope and* $\{(x,w) \in \text{Vert}(Q) : w > 0\}$ *is in bijection with* $\text{Vert}(P)$*, where the bijection is given by the map* $(x,w) \to x$*.*

Proof. It is clear that Q is bounded, since the coordinates are non-negative and $\leq \Gamma$. We note that Q is linearly isomorphic to $Q' = \{x \in \mathbb{R}^n : Ax = b, x_i \leq \Gamma \text{ for } i = 1,2,3,\ldots,n\}$ via the projection $\pi(x,w) = x$. Moreover, the vertices of Q with $w > 0$ map under π to the vertices of Q' with $x_i < \Gamma$ for all i. Therefore, as all the vertices of P have $\max_i x_i < \Gamma$ and every $v \in \text{Vert}(Q') \setminus \text{Vert}(P)$ has $\max_i v_i = \Gamma$, the vertices of Q' with $\max_i x_i < \Gamma$ are the vertices of P. □

E.2 Concluding the reduction

Theorem E.4 (Translation of Theorem 10 from Reimers and Arne)**.** *RVP is NP-hard on polytopes with branchwidth* ≤ 4*.*

Proof. Given an instance of SCIPF, $ax = b$, we construct $P(a,b)$ as in definition E.1. By theorem E.1, the existence of a vertex of P with $u > 0$ is equivalent to the existence of a solution $x \in \{0,1\}^n$ with $ax = b$. We construct the polytope Q as in appendix E.1, and by that lemma it follows that Q has a vertex with $u, w > 0$ iff P has a vertex with $u > 0$. Thus, that instance of SCIPF has a solution iff the instance of the restricted vertex problem given by $(P, \{u, w_1, \ldots, w_n\})$ has a solution.

Since Γ is has bit length polynomial in the binary encoding of A, b, the construction of the matrix representation of $Q(a,b)$ from (a,b) is polynomial time. Thus solving RVP on polyhedra of the form $Q(a,b)$ with $I = \{u, w_1, \ldots, w_n\}$ is NP-hard. As the matrix defining Q is obtained by a sequence of series extensions of the matrix defining P, it follows by lemma 6.1.8 that $\text{bw}(Q) \leq \max(\text{bw}(P), 2) \leq 4$.Therefore, solving RVP on polytopes with $\text{bw} \leq 4$ is NP-hard. □

Remark E.2.1. *The reduction in [35] only needs to introduce a single new variable, instead of* n *of them, because of the equations defining* P*. Otherwise it is essentially the same.*

F Branchwidth of the Circulation Polytope

As mentioned in the introduction, [27] proves that sampling vertices is NP-hard by using the circulation polytope of a directed graph. We will review this construction and argument below. One could wonder whether circulation polytopes are already of small branchwidth. In this section we verify that they are not, without further restrictions on the class of graphs. Additionally, we show that the vertex sampling problem is fixed parameter tractable in the branchwidth when restricted to class of circulation polytopes.

Definition F.0.1 (Incidence Matrix)**.** *Let $G = (V, E)$ be a directed graph. Let D be a matrix with columns indexed by E and rows indexed by V, where $D_{v,e} = 1$ if $e = (v, *)$, $D_{v,e} = -1$ if $e = (*, v)$, where $*$ denotes any vertex, and $D_{v,e} = 0$ otherwise. D is called the incidence matrix of G. When the graph is not clear from the context, we denote the incidence matrix of G by $D(G)$.*

Definition F.0.2 (Circulation Polytope)**.** *Given a directed graph G, with divergence operator* $\mathrm{div} : \mathbb{R}^E \to \mathbb{R}^V$,

$$\mathrm{div}(f)(v) = \sum_{e \in v^{in}} f(e) - \sum_{e \in v^{out}} f(e),$$

the circulation polytope is:

$$P(G) = \{x \in \mathbb{R}^E : x \geq 0, \sum x = 1, \mathrm{div}(x) = 0\}.$$

Since the divergence operator is given by $D(G)$, an equivalent description is $P(G) = P(\binom{\mathbb{1}}{D(G)}, e_1)$, where $\mathbb{1}$ is the all ones vector. We call the $x \in P(G)$ normalized circulations of G; circulations of G are $x \in \mathbb{R}^E$ with $x \geq 0$ and $\mathrm{div}(x) = 0$.

We call the set of such polytopes, presented in equational form using incidence matrix of a directed graph, the set of circulation polytopes. Again, note that the matrix of the presentation is part of our definition, we do not consider the question of recognizing whether the underlying set of a polytope can be presented as a circulation polytope. A theorem in [27] uses the circulation polytope to prove that sampling vertices of a polytope is hard, using [25][Proposition 5.1] and the following folklore theorem.

Theorem F.1 (Folklore, [27])**.** *For G any directed graph, the vertices of $P(G)$ are bijection with the simple cycles of G. Moreover, this correspondence is efficiently computable.*

Proof. The correspondence sends a directed simple cycle C to $x_C := \frac{1}{C} 1_C$, and maps a vertex to its support. Below we will use the following facts: 1) There is only one normalized circulation of a directed simple cycle. 2) An acyclic digraph has no nonzero circulations. The proof now follows:

- First, we verify that every directed simple cycle produces a vertex. Let C be a simple cycle, and suppose that there are x, x' so that $x_C = \frac{x+x'}{2}$. Then, because $x, x' \geq 0$, $\mathrm{supp}(x_C) = \mathrm{supp}(x) \cup \mathrm{supp}(x')$. Thus, x and x' are normalized circulations on the directed simple cycle graph C, hence $x = x'$.
- Next, we verify that the support of every vertex x is a directed simple cycle. We will suppose for contradiction that $\mathrm{supp}(x)$ is not a directed simple cycle, first argue that there are simple cycles $C \neq C'$ such that $C, C' \subseteq \mathrm{supp}(x)$, and then use C, C' to write x as an average of distinct circulations, showing that it cannot be a vertex.
  First, we observe that there is a simple cycle $C \subseteq \mathrm{supp}(x)$, since any acyclic digraph has no nonzero circulations. Let $m = \min_{e \in C}(x(e))$, and suppose that e is an edge with $x(e) = m$. We have that $x - m1_C$ satisfies $\mathrm{div}(x - m1_C) = 0$ and $x - m1_C \geq 0$. By assumption $x \neq 0$, so $y = \frac{1}{\sum_f (x - m1_C)(f)}(x - m1_C) =: y$ is a circulation. Thus, $\mathrm{supp}(y)$ has a nonzero circulation, and must therefore contain a directed simple cycle, say C'. However, $C' \neq C$, since $e \notin C'$ as $y(e) = \frac{1}{\sum_f (x - m1_C)(f)}(x(e) - m1_C(e)) = 0$.
  Given C and C' as in the previous paragraph, we define $x^+ = x + \frac{\epsilon}{C} 1_C - \frac{\epsilon}{C'} 1'_C$ and $x^- = x - \frac{\epsilon}{C} 1_C + \frac{\epsilon}{C'} 1'_C$. For $\epsilon > 0$ we have $x^+ \neq x^-$ and $x = \frac{x^+ + x^-}{2}$ and $\mathrm{div}(x^+) = \mathrm{div}(x^-) = 0$. Additionally, for ϵ sufficiently small, we have $x^+, x^- \geq 0$, hence $x^+, x^- \in P(G)$, and so x could not be a vertex. We turn to establishing the existence of C, C'.

□

This, along with proposition 5.1 of [25], which shows that sampling directed simple cycles is NP-hard, gives the proof [27]:

Theorem F.2 ([27])**.** *Sampling vertices is NP-hard, in the sense that a polynomial time uniform sampler would imply that* RP = NP.

Since the focus of this paper is to determine the impact of polytope branchwidth on the problem of sampling vertices, it is natural to ask about the branchwidth of the circulation polytopes. Indeed, if the branchwidth of circulation polytopes was already bounded, Khachiyan's proof that sampling vertices was NP-hard would extend to bounded branchwidth polytopes. However, we will show that circulation polytopes have unbounded branchwidth and that the vertex sampling problem is tractable on circulation polytopes of bounded branchwidth. To do all this, we will need to recall the notion of branchwidth of an undirected graph.

Definition F.0.3 (The branchwidth of an undirected graph - see [20])**.** *We use the definitions and notation about labelled cubic trees from §6. Consider an undirected graph $G = (V, E)$. An E-labelled cubic trees (T, ν) is called a branch decomposition of G. For a subset $J \subseteq E$, let $G[J]$ denote the subgraph induced by J, and for a graph $G = (V, E)$ let $V(G) = V$ denote the set of edges of G. The width of an edge f in T is $\lambda_T(f) = |V(G[L_f]) \cap V(G[R_f])|$, and the width of T is $\max_{f \in T} \lambda_T(f)$. The branchwidth of G is the minimum width of a branch decomposition of G.*

We also need:

Definition F.0.4 (Branchwidth of a matroid - see [20])**.** *(We use the definitions and notation from §6.) Consider a matroid $M = (E, \mathcal{I})$ with rank function ρ and ground set E. A branch decomposition is an E-labelled cubic tree (T, ν). The width of an edge f in T is $\lambda_T(f) = \rho(L_f) + \rho(L_r) - \rho(M) + 1$, and the width of T is $\max_{f \in T} \lambda_T(f)$. The branchwidth of M is the minimum width of a branch decomposition of M.*

This comports with the concept of branchwidth used in this paper:

Lemma F.0.5. *If A is a matrix, and $M(A)$ denotes the matric matroid on A, then $\mathrm{bw}(M(A)) = \mathrm{bw}(P(A, b))$, for any vector b.*

Proof. This follows because the matroid rank function is the linear algebra rank of the submatrix formed by the corresponding columns. □

We are interested in the directed simple cycles of a directed graph, but the theory about branchwidth and fixed parameter tractability of graph algorithms we will use is stated in terms of undirected graphs. We will therefore need the following definition in order to relate the two directed and undirected settings:

Definition F.0.6 (Underlying undirected graph)**.** *Given a directed graph $G = (V, E)$, we let $\overline{G}$ denote the underlying undirected graph: the nodes of $\overline{G}$ are the same as the nodes of G, and there is an undirected edge $\{a, b\}$ in $\overline{G}$ if and only if $(a, b) \in E \vee (b, a) \in E$.*

We recall:

Theorem F.3 (Theorem 4 of [20], see also [32] for a related result)**.** *Let $G = (V, E)$ be any undirected graph that has a cycle. Then $\mathrm{bw}(M(G)) = \mathrm{bw}(G)$, where $M(G)$ is the cycle matroid of G, i.e. the matric matroid of $D(G)$.*

This has the following corollary:

Corollary F.3.1. *Let $G = (V, E)$ be a directed graph that has a cycle and that has no digons (edges e, e' with $e = (a, b)$ and $e' = (b, a)$). Then $\mathrm{bw}(P(G)) = \mathrm{bw}(P(\begin{pmatrix} \mathbb{1} \\ D(G) \end{pmatrix}, e_1)) \geq \mathrm{bw}(\overline{G})$.*

Proof. As G has no digons, $D(G)$ is an oriented incidence matrix of $\bar{G}$. Thus, the matroids $M(\bar{G})$ and the matric matroid of $D(G)$ are equal. This implies that $\mathrm{bw}(D(G)) = \mathrm{bw}(M(\bar{G})) = \mathrm{bw}(\bar{G})$. Thus, by lemma D.0.2, we have $\mathrm{bw}(\begin{pmatrix} \mathbb{1} \\ D(G) \end{pmatrix}) \geq \mathrm{bw}(D(G)) - 1 = \mathrm{bw}(\overline{G}) - 1$.

□

We now recall:

Theorem F.4 ([16, 36])**.** *Let G be an $n \times n$ grid graph. Then $\mathrm{bw}(G) = n$.*

Corollary F.4.1. *Let H be an $n \times n$ grid graph, and let G be any digraph with $\overline{G} = H$ and with no digons. Then the circulation polytope $P(G)$ has $\mathrm{bw}(P(G)) \geq n - 1$.*

Remark F.0.7. *The condition about digons can be removed, they just make the arguments cleaner.*

Thus, the branchwidths of circulation polytopes are unbounded. To wrap up this section, we will prove that if we only consider circulation polytopes, then the vertex sampling problem is in FPT in the branchwidth.

We recall the notion of treewidth. An equivalent, and in some respects conceptually simpler definition, can be found following definition F.2.1 below:

Definition F.0.8 (Treewidth [19])**.** *Let $G = (V, E)$ be a graph. A tree decomposition of G is a pair $(\{X_i : i \in I\}, T = (I, M))$ where $\{X_i : i \in I\}$ is a collection of subsets of V, called bags, and T is a tree such that:*

- $\bigcup_{i \in I} X_i = V$
- *$(u, v) \in E$ implies there is some $i \in I$ with $u, v \in X_i$.*
- *For all vertices v, $\{i \in I : v \in X_i\}$ induces a connected subtree of T.*

The width of a tree decomposition $(\{X_i : i \in I\}, T = (I, M))$ is $\max_{i \in I} |X_i| - 1$. The treewidth of G, $\mathrm{tw}(G)$ is the minimum width over every possible tree decomposition of G.

Theorem F.5 (Theorem 5.1 of [36])**.** *For any graph G, $\mathrm{tw}(G) \leq \max(\lfloor \frac{3}{2}\mathrm{bw}(G) \rfloor, 2) - 1$.*

Proposition F.0.9. *Fix an integer k. Then, there is a polynomial time algorithm to uniformly sample from the vertices of any circulation polytope of branchwidth $\leq k$.*

Proof. If $P(G)$ is such a circulation polytope, it would suffice to uniformly sample from the directed simple cycles of G because of the efficiently computable bijection given by theorem F.1. We can assume that G has a cycle, otherwise the polytope is empty and the sampling problem is trivial. Recall that $\overline{G}$ denotes the undirected simple graph underlying G.

Construct G' from G by removing one edge from each digon, then by lemma 6.1.1, $\mathrm{bw}(D(G')) \leq \mathrm{bw}(D(G))$. Since the underlying undirected simple graphs of G and G' are equal, $\mathrm{bw}(\overline{G}) = \mathrm{bw}(D(G'))$ by theorem F.3. Finally, we use that $\mathrm{bw}(D(G)) \leq \mathrm{bw}(\binom{\mathbb{1}}{D(G)}) = \mathrm{bw}(P(G))$ to conclude that $\mathrm{bw}(\overline{G}) = \mathrm{bw}(D(G')) \leq \mathrm{bw}(D(G)) \leq \mathrm{bw}(P(G)) \leq k$. Thus, $\overline{G}$ has treewidth $\leq \max(\lfloor \frac{3}{2}k \rfloor, 2) - 1$ by theorem F.5. In appendix F.1 we verify that the problem of sampling directed simple cycles is fixed parameter tractable in the treewidth of the underlying undirected graph, proving the claim. □

F.1 Counting directed simple cycles for bounded treewidth graphs

To finish the arguments in proposition F.0.9, we verify that sampling directed simple cycles is fixed parameter tractable in the the treewidth of the underlying undirected graph. We do this by encoding it into a related problem about undirected graphs, after which we can appeal to metatheorems about MSO_2.

Definition F.1.1 (Spoon Representation of a Digraph)**.** *Let $G = (V, E)$ be a directed graph. We construct an undirected representation of it in the following way (probably most clearly understood from fig. 5): First, start with the node set V and no edges. Then, for each directed edge $(a, b) \in E$, add 2 nodes $[ab]$ and $[ab]'$ to V, and add undirected edges: $a, [ab], [ab], b, [ab], [ab'], [ab'], b$, as in fig. 5i). Call the resulting graph $s(G)$; the spoon graph associated to G. The spoon graph associated to a single edge, depicted in fig. 5i), is called a spoon graph.*

Instead of directed simple cycles in G, we will look for spoon cycles in $s(G)$.

Definition F.1.2 (Spoon Cycle)**.** *Let C be a directed cycle graph of length k. Then, a spoon cycle of length k is a graph isomorphic to $s(C)$, the spoon graph associated to C. See fig. 5ii).*

Lemma F.1.3 (Equivalence between directed simple cycles and spoon cycles)**.** *The number of directed simple cycles in G is the same as the number of spoon cycles of $s(G)$.*

Proof. Let $D(G)$ denote the set of directed simple cycles of G, and let $S(s(G))$ denote the set of spoon cycles of $s(G)$. We define a map $sp : D(G) \to S(s(G))$ that maps a directed cycle to a spoon cycle by mapping the set of edges to the set of spoons corresponding to those edges. In the other direction, we define $e : S(s(G)) \to D(G)$ by mapping the set of spoons to the set of edges corresponding to those spoons. These two maps are inverses. □

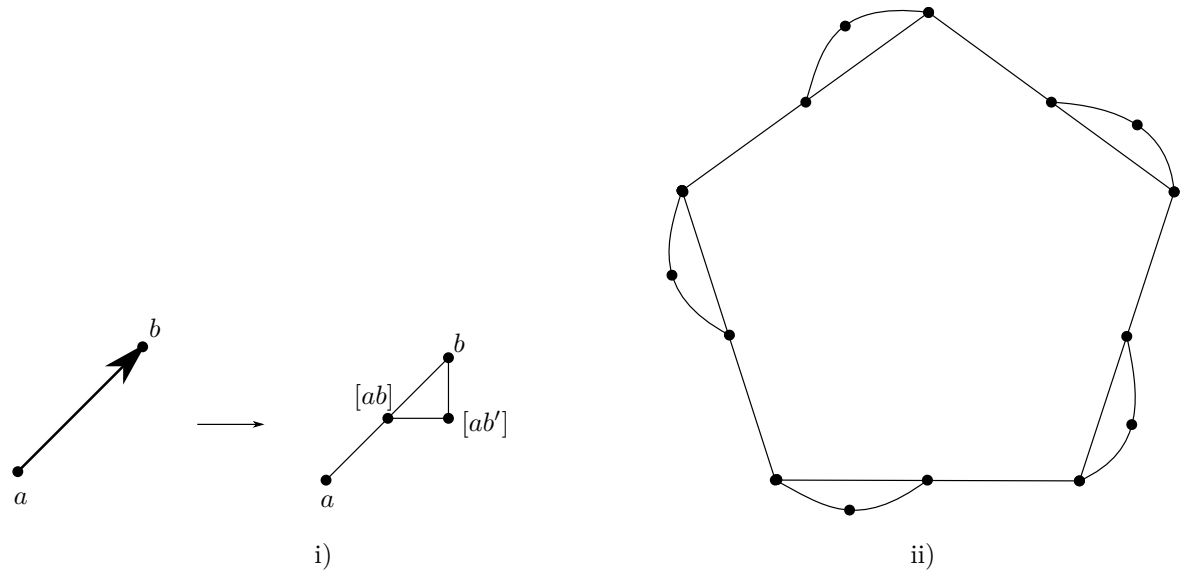

Figure 5: Spoon Constructions: i) Replacing a directed edge with a spoon. ii) A spoon cycle of length 5.

Next we will show that there is a formula in monadic second order logic for graphs[5], MSO_2, that characterizes the spoon cycles of a given graph. To do so, we need to characterize spoon cycles in terms of properties that are more easily expressed in MSO_2. The following definition and lemma will make the proof of the characterization cleaner:

Definition F.1.4 (Coning off an edge)**.** *If e is an edge in a graph G, we will say that the graph G' is obtained by coning off the edge e if $V(G') = V(G) \cup \{*\}$, and $E(G') = E(G) \cup \{\{*, a\}, \{*, b\}\}$ where $*$ is some vertex not in G. If $J \subset G$, we say that G' is obtained from G by coning of J if G' was obtained by iteratively coning off the edges in J.*

Lemma F.1.5. *If G is a simple cycle of length $2k$, and $J \subseteq E$ is a maximal set of non-pairwise adjacent edges, then if G' is obtained from G by coning off the edges in J, then G' is a spoon cycle of length k.*

Now, we have the characterization of spoon cycles:

Lemma F.1.6. *Let H be a non-empty graph. Then H is a spoon cycle with at least 2 spoons iff:*

1. *H is connected.*
2. *The degrees of nodes in H are either 2 or 3.*
3. *Both neighbors of every degree 2 node in H have degree 3 and are connected to eachother.*
4. *The neighbors of degree 3 nodes have degree 2, 3 and 3.*

Proof. Let $Z = \{v \in H : \deg(v) = 2\}$. Let $H' = H \setminus Z$, that is, the graph obtained by deleting all vertices in Z, and incident edges. Then, by 3., H' remains connected. Since Z^c in H consists of degree 3 nodes, by 2., and each node in Z^c is incident to one node in Z by 4, the nodes in Z^c all have degree 2 in H'. Thus, H' is a simple cycle.

It remains to check that H is obtained from H' by coning off a maximal set of non-pairwise adjacent edges in H'. Let us consider the set of edges $J = \{\{a, b\} : \exists v \in Z, v \sim a, v \sim b\}$. We will show that J is a maximal set of non pairwise incident edges in the cycle H, that H is obtained from H' by coning off all the edges in J, and that H has $2k$ edges. Taken together, those prove via lemma F.1.5 that H is a spoon cycle. We now verify those properties of J and H.

First, we check that J consists of a non-pairwise adjacent edges of H'. Suppose that $e = \{a, b\}, e' = \{b, c\} \in J$. Then there are $v, v' \in Z$ with $v \sim a, v \sim b$ and $b \sim v', c \sim v'$ in H. However, this implies that

[5]For background on second order logic, the reader is referred to [11, Chapter 7]; for background on these meta-theorems and MSO_2, the reader is referred to [4].

$\{a, v, v', c\}$ are all neighbors of b in H. Since the max degree is 3, some pair of those have to be equal. We know that $a \neq c$, otherwise $e = e'$, and that $\{a, c\} \cap \{v, v'\} = \emptyset$ as the former is in Z^c and the latter in Z. Therefore, the only possibility is that $v = v'$. However, this would imply that v is adjacent to $a \neq b \neq c$, which contradicts $v \in Z$.

Next, we check that H has an even number of edges, and that J is a maximal set. Suppose that $(w_1, w_2, w_3, w_4, w_5, w_6)$ forms a path in H, and suppose that $\{w_3, w_4\} \notin J$. We will argue that $\{w_1, w_2\} \in J$, and the same argument will show that $\{w_5, w_6\} \in J$. Therefore, since we already know that edges in J cannot be adjacent, every other edge in H is in J, and in particular H has an even number of edges. Since $w_3 \notin Z$, it had degree 3 in H. Say that $v \in Z$ is incident to w_3 in H. We have that v is incident to either w_2 or w_4. However, since $\{w_3, w_4\} \notin J$, v cannot be incident to w_4. Hence, v is incident to w_2, implying that $\{w_2, w_4\} \in J$.

Finally, we observe that by the definition of J and Z, H is obtained from H' by coning off all the edges in J.

□

Lemma F.1.7. *There is an MSO_2 formula for spoon cycles.*

Proof. The formula will be over a subset of the edges, and will be obtained by ANDing together formulas for each of the 4 bullets in lemma F.1.6.

First, here is an MSO_2 formula that checks that a node v has degree $\geq k$ in the graph induced by $J \subseteq E$: $\deg_{\geq k,J}(v) := \exists_{v_1,\ldots v_k \in V} \wedge \exists_{e_1,\ldots,e_2 \in J}(\bigwedge_{i=1,\ldots k} \text{inc}(v, e_i) \wedge \bigwedge_{i=1,\ldots,k} \text{inc}(v_i, e_i) \wedge (\bigwedge_{1 \leq i < j \leq k} v_i \neq v_j)$. Thus, $\deg_{k,J}(v) = \phi_{\geq k,J}(v) \wedge \neg\phi_{\geq k+1,J}(v)$ expresses that v has degree exactly k.

Second, as a shorthand for referring to a property ψ quantified over nodes in an edge induced subgraph, we define $Q_{v \in G[J]}\psi(v) = Q_{v \in V}\exists_{e \in J}\text{inc}(v, e) \wedge \psi(v)$, where $Q \in \{\exists, \forall\}$. We also define $\text{inc}_J(a, b) = \exists_{e \in J}\text{inc}(a, e) \wedge \text{inc}(b, e)$. Given this shorthand notation, here are MSO_2 expressions for the 4 properties of lemma F.1.6:

1. $\forall_{Y \subseteq V}((\forall_{s \in Y}\exists_{e \in J}\text{inc}(s, e)) \wedge [[\exists_{u,v \in V}\exists_{e' \in X} \wedge \text{inc}(e', v) \wedge u \in Y \wedge v \notin Y] \rightarrow \exists_{e'' \in J}\exists_{u' \in Y, v' \notin Y}\text{inc}(u', e'') \wedge \text{inc}(v', e'')])$.
   We clarify that this formula checks that for any non-empty proper subset of vertices, Y, in the subgraph $G[J]$, that there is an edge in J going from Y to the complement in Y in $G[J]$. Thus, this expression is true iff $G[J]$ is a connected subgraph.
2. $\forall_{v \in G[J]} \deg_{2,J}(v) \vee \deg_{3,J}(v)$.
3. $\forall_{v,a,b \in G[J]}(\deg_{2,J}(v) \wedge \text{inc}_J(v, a) \wedge \text{inc}_J(v, b) \wedge a \neq b) \rightarrow (\text{inc}_J(a, b) \wedge \deg_{3,J}(a) \wedge \deg_{3,J}(b))$.
4. $\forall_{v,x_1,x_2,x_3 \in G[J]}(\deg_{3,J}(v) \wedge \bigwedge_{i \in [3]} \text{inc}_J(v, x_i) \wedge \bigwedge_{i,j \in [3], i \neq j} x_i \neq x_j)) \rightarrow (\bigvee_{\sigma \in S_3}(\deg_{3,J}(x_{\sigma(1)}) \wedge \deg_{3,J}(x_{\sigma(2)}) \wedge \deg_{2,J}(x_{\sigma(3)})))$

□

Thus, we have the following:

Theorem F.6 (Counting directed simple cycles)**.** *The problem of counting directed simple cycles of a digraph G is fixed-parameter tractable in the treewidth of the underlying undirected, simple graph $\overline{G}$.*

Proof. First, we note that $\text{tw}(s(G)) = \max(2, \text{tw}(\overline{G}))$, using proposition F.2.5. Now the result follows from lemma F.1.3, lemma F.1.7 and the main theorem of [4]. □

Now we prove the main theorem of this section:

Proposition F.1.8. *There is a polynomial time algorithm for uniformly sampling directed simple cycles is from directed graphs where the underlying undirected graph has bounded treewidth.*

Proof. Let $DSC(G)$ denote the set of directed simple cycles of a graph $G = (V, E)$. It suffices to show that for any sets $I, J \subseteq E$, one can calculate $\mu_{I,J} := |\{\alpha \in DSC(G) : I \subseteq \alpha, J \cap \alpha = \emptyset\}|$ (see [34] Appendix B1 for details on how to sample given such marginals). Note by lemma F.2.4 that replacing edges by chains of bigons does not increase the treewidth of the underlying graph, provided it is already 2, since the effect on the underlying graph is just a sequence of series extensions. Similarly, deleting edges cannot increase the treewidth. Given sets of edges, I, J, we let $G_{d,I,J}$ denote the graph with the edges in J removed, and

the edges in I replaced by a chain of d bigons. We have that $|DSC(G_{d,I,J})| = \sum_{k=0}^{|I|} 2^{dk} |\{\alpha \in DSC(G) : |I \cap \alpha| = k, J \cap \alpha = \emptyset\}|$. Thus, calling a machine to count $|DSC(G_{d,I,J})|$ at $d = 1, 2, \ldots, |I|$ lets us compute $\mu_{I,J} = |\{\alpha \in DSC(G) : |I \cap \alpha| = |I|, J \cap \alpha = \emptyset\}|$ by inverting the corresponding Vandermonde system. □

F.2 Series-parallel extensions do not increase the tree width past 2

The arguments in this section are well-known, the author just couldn't find a reference for them. To streamline the arguments, we recall the notion of a partial k-tree:

Definition F.2.1 (k-trees, partial k-trees)**.** *A k-tree is any graph that can be recursively constructed in the following manner. We start with a tree T_0 that is a k clique. Then, we obtain T_n from T_{n-1} by picking any k-clique Q of T_{n-1} adding a new vertex v and connecting v to each node of Q. A* partial k*-tree is any subgraph of a k-tree.*

This provides an equivalent definition to the notion of treewidth used in [36] by the following:

Proposition F.2.2 (Theorem 4.4 of [19])**.** *G is a partial k-tree if and only if* $\mathrm{tw}(G) \leq k$*. Thus,* $\mathrm{tw}(G) = \min\{k : G \text{ is a partial } k\text{-tree}\}$.

We use the following lemmas above.

Lemma F.2.3 (Split-parallel edge extension)**.** *Let $G = (V, E)$ be a graph, and let $e = \{a, b\} \in E$. Let G' be obtained from G by adding a new vertex v, along with edges $\{a, v\}$ and $\{v, b\}$, which we will call a split-parallel edge extension. Then* $\mathrm{tw}(G') \leq \max(\mathrm{tw}(G), 2)$

Proof. Suppose that $\mathrm{tw}(G) = k \geq 2$. Then there is a k tree H such that G is a subgraph of H. Every edge in H is in some clique, as can be seen from the inductive construction of k-trees. Thus, let K be a clique containing $e = \{a, b\}$, and let G' be obtained by coning K with a cone vertex v. Then we obtain an embedding of H' by using the vertex v as well. Thus, $\mathrm{tw}(G') \leq k$, which implies the result. On the other hand, if $\mathrm{tw}(G) = 1$, then G is a forest and in particular embeds into a 2-tree, at which point we can repeat the above argument. □

Lemma F.2.4 (Series extension)**.** *Let G be a graph. Let G' be obtained by a series extension splitting $e \in E$ into e_1, e_2. Then* $\mathrm{tw}(G') \leq \mathrm{tw}(G)$.

Proof. Suppose that $\mathrm{tw}(G) = k \geq 2$. Then there is a k tree H such that G is a subgraph of H. Every edge in H is in some clique, as can be seen from the inductive construction of k-trees. Thus, let K be a clique containing $e = \{a, b\}$, and let G' be obtained by coning K with a cone vertex v. Then we obtain an embedding of H' be rerouting e_1 and e_2 as (a, v, b). Thus, $\mathrm{tw}(G') \leq k$, which implies the result. On the other hand, if $\mathrm{tw}(G) = 1$, then G is a forest and so is G'. □

Together, these imply:

Proposition F.2.5 (Treewidth of underlying undirected and spoon graph)**.** *Let G be a directed graph. Let $\overline{G}$ be the underlying undirected graph, and let $s(G)$ be the spoon graph. Then* $\mathrm{tw}(s(G)) \leq \max(2, \mathrm{tw}(\overline{G}))$.

Proof. One can obtain $s(G)$ from $\overline{G}$ by series and split-parallel extensions, and therefore the result follows from lemma F.2.4 and lemma F.2.3. See fig. 6 and the caption for an explanation of how to obtain $s(G)$ from $\overline{(G)}$.

□

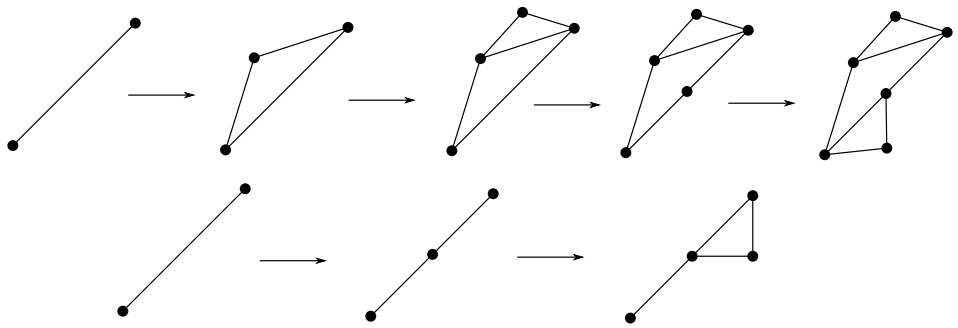

Figure 6: For any digon in G we replace the underlying edge in $\overline{G}$ by a digon of spoons. This can be done by applying a sequence of series and parallel-split extensions by following the local instructions of the top image. For every directed edge of G which is not part of a digon we apply the series extension then the split parallel edge extension to obtain a spoon, following the local instructions of the bottom image.