COMPLEXITY AND GEOMETRY OF SAMPLING CONNECTED GRAPH PARTITIONS

LORENZO NAJT*, DARYL DEFORD†, JUSTIN SOLOMON†

Abstract. In this paper, we prove intractability results about sampling from the set of partitions of a planar graph into connected components. Our proofs are motivated by a technique introduced by Jerrum, Valiant, and Vazirani. Moreover, we use gadgets inspired by their technique to provide families of graphs where the "flip walk" Markov chain used in practice for this sampling task exhibits exponentially slow mixing. Supporting our theoretical results we present some empirical evidence demonstrating the slow mixing of the flip walk on grid graphs and on real data. Inspired by connections to the statistical physics of self-avoiding walks, we investigate the sensitivity of certain popular sampling algorithms to the graph topology. Finally, we discuss a few cases where the sampling problem is tractable. Applications to political redistricting have recently brought increased attention to this problem, and we articulate open questions about this application that are highlighted by our results.

1. Introduction. The problem of *graph partitioning*, or dividing the vertices of a graph into a small number of connected subgraphs that extremize an objective function, is a classical task in graph theory with application to network analytics, machine learning, computer vision, and other areas. Whereas this task is well-studied in computation and mathematics, a related problem remains relatively understudied: understanding how a given partition compares to other members of the set of possible partitions. In this case, the goal is not to generate a partition with favorable properties, but rather to compare a given partition to some set of alternatives. Recent analysis of political redistricting have invoked such comparisons (see §1.3), motivating the investigation of this general problem.

Consider a connected graph $G = (V, E)$, and let $P_k(G)$ denote the collection of k-partitions of V such that each block induces a connected subgraph. One approach to understanding how a given element of $P_2(G)$ compares to the other elements proceeds by uniformly sampling from $P_2(G)$, then gathering statistics about this sample and comparing them to the partition under consideration.While this is an attractive approach, this uniform sampling problem is computationally intractable, assuming NP $\neq$ RP.

We open the paper by reviewing this fact. Our first new result is that this intractability persists even if we consider partitions of equal size. Then, motivated to produce a result that is more relevant to the classes of graphs that arise in redistricting, we show that uniformly sampling $P_2(G)$ remains intractable even if G is a maximal plane graph with a constant bound on the vertex degree. Beyond sampling from the uniform distribution, we also prove results about the intractability of sampling from a broader class of distributions over connected k-partitions.

Such worst case results should not be considered proof that uniformly sampling from $P_k(G)$ is *always* impossible. However, it does indicate that algorithm designers should examine sampling heuristics with some skepticism. Driven by this philosophy, we follow up our investigation of the worst-case complexity with an investigation into applicability of a general and often extremely useful sampling tool, which is Markov chain Monte Carlo.

In the context of redistricting, Markov chains have rapidly become a popular tool for sampling from $P_k(G)$ to compare a districting plan (§1.3) to the space of possible plans [16,29,31,77,80]. The most commonly used Markov chain moves randomly on $P_2(G)$ by proposing to change the block assignment of a uniformly chosen node, and accepting such moves only if the connectivity of each block is preserved [16, 31]. We call this chain the flip walk (Definition 3.1). If G is 2-connected, then the flip walk on $P_2(G)$ is irreducible, and the stationary distribution is uniform [12]. In principle, running the flip walk on $P_2(G)$ for a long time will produce a uniformly random element of $P_2(G)$. For this approach to be computationally feasible, however, one must guarantee that the mixing time of the Markov chain on $P_2(G)$ is not too large compared to $|G|$.

Pursuing this angle, we explain how to engineer a family of graphs $G \in \mathcal{G}$ so that the mixing time of the flip walk on $P_2(G)$ grows exponentially quickly in $|G|$. Based on this, as well as some empirical work motivated by the bottlenecks we discover, we can conclude that there are strong reasons to doubt that Markov chain methods based on the flip walk mix in polynomial time.

*University of Wisconsin-Madison. Corresponding Author: Lnajt@math.wisc.edu
†Massachusetts Institute of Technology.

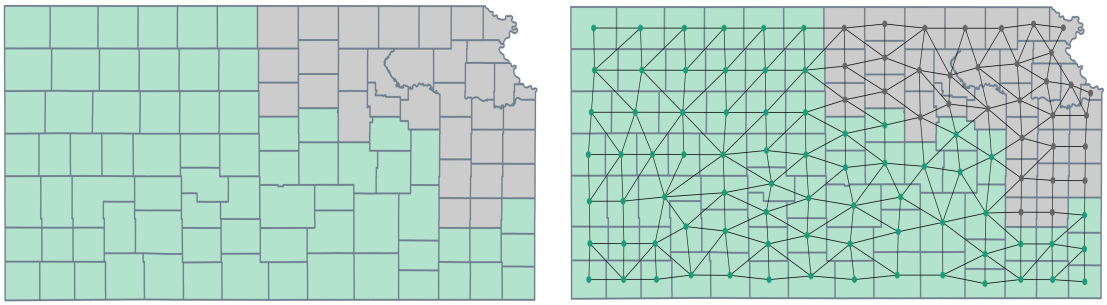
In addition to this, we make a connection with the literature on self-avoiding walks [43] that demonstrates the existence of dramatic phase transitions in the qualitative behavior of distributions on $P_2(G)$. We provide experiments illustrating the relevance of these phase transitions to redistricting and, inspired by the ideas in those experiments, we examine the robustness of other popular approaches to sampling from $P_2(G)$, including some methods based on spanning trees [37, 41]. Overall, the observations we make highlight interesting and difficult challenges for the sampling algorithms and inference principles being used in statistical analysis of redistricting plans.

Finally, we discuss a few classes of graphs on which it is possible to sample uniformly from $P_2(G)$ in polynomial time, but which are far from the kinds of graphs relevant to redistricting. The large gap between where we know that uniform sampling is intractable and where we know it is tractable, along with some connections to outstanding problems from statistical physics that seem to be on par with the intended redistricting application, indicates that there are many challenging questions remaining about sampling from $P_k(G)$.

Overview and contributions. As part of a broader effort to establish mathematical underpinnings for the analytical tools used in redistricting [12, 13, 31, 67, 77], we identify challenges and opportunities for further improvement related to random sampling in the space of graph partitions. In addition to the technical material listed below, we articulate some implicit assumptions behind outlier methods used in the analysis of gerrymandering (§6.2.2), and offer some suggestions for future work around computational redistricting.

- Sampling intractability results, and bottlenecks:
  - We review why it is intractable to sample uniformly from $P_2(G)$ (§2.3). As is typical, our strategy will be to engineer graphs so that uniform samples from their connected 2-partitions are likely to solve an NP-hard problem. We will work with planar graphs to leverage bond-cycle duality, which gives a bijection between $P_2(G)$ and the set of simple cycles of the dual.
  - One realistic condition to put on samplers from $P_2(G)$ is to restrict to the set of *balanced* partitions, partitions for which both blocks have the same number of nodes. We next show that uniformly sampling balanced 2-partitions remains NP-hard (§2.4).
  - We also prove the intractability of uniformly sampling from $P_2(G)$ for an even more constrained family of graphs: planar triangulations with bounded vertex degree (§2.5).
  - We prove that uniformly sampling k-partitions is intractable, using a generalization of bond-cycle duality (Appendix A.3).
  - The gadgets used in the intractability proofs provide a means for constructing families of graphs such that the flip walk Markov chain on $P_2(G)$ has exponentially large mixing time (§3). We describe a family of plane triangulations with vertex degree bounded by 9 such that the flip walk on $P_2(G)$ mixes torpidly.
- Empirical results:
  - We include some empirical evidence indicating that Markov chains based on the flip walk mix slowly on grid graphs (§4.2) and on the graphs used in analysis of redistricting (§4.2.1).
  - We mention a link between our sampling problem on grid graphs and long standing challenges regarding the self-avoiding walk model from statistical physics (§4.1). We use this connection to motivate and demonstrate phase transitions in the qualitative properties of $P_2(G)$.
  - We provide experiments (§4.3.1 and §4.3) demonstrating that popular methods for sampling from geographic partitions via discretizing the geography as a graph G and sampling from $P_2(G)$ are impacted in surprising ways by the discretization used.
- Positive results:
  - We prove that there are efficient and implementable dynamic programming algorithms that can be used to sample uniformly from $P_2(G)$ and to sample uniformly from the balanced partitions in $P_2(G)$, provided that G is a series-parallel graph. This algorithm succeeds in some cases where the flip walk is unreliable. We observe that these sampling problems are tractable on graphs of bounded treewidth.

1.1. Related Work. We are not the first team of researchers to have considered redistricting problems from a complexity point of view. Indeed, there are many papers showing that optimization problems related to designing the most "fair" or "unfair" districts are NP-hard, for various meanings of the word fair; works in this category include [13, 67]. Other researchers have explored the complexity of findings paths through



a) b)

Figure 1.1: a) Kansas with county units [25], along with a connected 2-partition. b) The corresponding state dual graph overlayed. Much more granular subdivisions of a state are often used.

the flip walk state space [12]. We will discuss other related work in the body of the paper, such as the connection to self-avoiding walks (§4.1), and related sampling problems (§3.1).

1.2. Basic Notation. Let $G = (V, E)$ be a graph; unless otherwise specified, all of our graphs will be undirected, finite and simple. If unspecified, usually $n := |V|$. Given a graph G, $V(G)$ denotes the set of nodes, and $E(G)$ the set of edges. An (ordered) *k-partition* $P = (V_1, V_2, \ldots, V_k)$ of G is an list of disjoint subsets $V_i \subseteq V$ whose union is V, while an unordered k-partition is a set $\{V_1, \ldots, V_k\}$ satisfying the same conditions. Throughout this paper we will be concerned with *connected k-partitions*, i.e., those k-partitions where each V_i induces a connected subgraph. The set of ordered connected k-partitions of G is denoted $P_k(G)$, and the set of unordered connected k-partitions of G is denoted $\mathscr{P}_k(G)$. If $A \subset V$, then we will use $\partial_E A$ to denote the *edge boundary* of A: $\partial_E A = \{\{u, v\} \in E : u \in A, v \notin A\}$.

1.3. Motivation from Redistricting. In the United States, states are divided into small geographical units, such as in Figure 1.1a); these units are combined into voting districts, each of which elects a single representative. An assignment of these units to a district is called a districting plan. The units can be represented by the nodes of a graph, where units that share common boundaries are adjacent, as in Figure 1.1. This graph is called the state dual graph. Assuming that voting districts must be contiguous, as is usually the case, a districting plan with k-districts is modeled by a connected k-partition of the state dual graph.

It was quickly observed [57, 76] that by a clever choice of districting plan one could engineer aspects of electoral outcomes, a practice known as "gerrymandering." In an effort to counteract this, there have been many proposals to design districting plans algorithmically, a process which often involves grappling with computationally intractable problems [13, 67]. The reality of redistricting, however, is that the power to draw the graph partition is in the hands of a legislature, dedicated committee, or hired expert—rather than a piece of software. For this reason, rather than using an algorithm to draw plans in the first place, some have suggested to analyze already drawn plans for compliance with civil rights law or desirability relative to alternatives.

Arguments for or against districting plans are facilitated by understanding a plan in the context of what is possible. For instance, an argument that a plan was drawn with the intent to discriminate might calculate that the proposed plan has more discriminatory properties than the vast majority of plans from a randomly generated collection of comparable plans; more specifically, the claim is that a particular map is an outlier compared to the other possibilities [28] (see also §6.2.2). This contextual approach requires sampling a *diverse ensemble* of plans that are compliant with the principles laid out by the governing body. A variety of algorithms have been proposed to sample ensembles of graph partitions for this purpose, from genetic algorithms [32, 71] to random walks [31, 54]; recent expert reports in redistricting cases have used these tools to generate quantitative assessments of proposed plans [29, 54, 55, 80].

While random walk methods like [54,77] are guaranteed to sample from an explicitly designed distribution if run for long enough, practical computational constraints make it impossible to reach that point if there

are no guarantees on the mixing time. On the other hand, algorithms like [30, 41, 75], which are intended to generate a diverse set of partitions, sample from unknown distributions whose properties are hard to characterize.

Thus, two critical open problems arise when relying on measurements derived from random ensembles of districting plans. First, it is difficult to verify whether ensemble generation algorithms produce a statistically-representative sample from a targeted distribution. We study this problem by asking whether certain distributions over partitions are efficiently sampleable (§2 and §5), and whether certain sampling algorithms run efficiently (§3 and §4). Second, the qualitative properties of distributions over partitions are challenging to characterize, and it is difficult to determine the extent to which an outlier classification is affected by modelling decisions, including the choice of sampling algorithm or discretization. We study this problem in §4.3 and §4.3.1.

2. Sampling Intractability. In this section, we present our results about the intractability of various general sampling problems associated with connected k-partitions. The key idea [63] behind proving that some uniform sampling problem is intractable is to show that one can modify an algorithm that solves it into an algorithm that samples from the solutions to some hard problem. We begin by setting up some language (§2.1), some of which is standard, and some of which we have created to organize our results. Then, we review the intractability of uniformly sampling from $P_2(G)$ (§2). Next, we show that uniformly sampling from balanced connected 2-partitions is intractable (§2.4). Then, we show that uniformly sampling from $P_2(G)$ remains intractable under certain constraints on the topology of G §2.5. Finally, we will show that, for any fixed k, certain generalizations of the uniform distribution on $P_k(G)$ are intractable to sample from (§2.6).

2.1. Preliminaries on Intractability of Sampling. In this section, we discuss some background on sampling problems, the class RP, why RP $\neq$ NP is a reasonable assumption, and what it means for a sampling problem to be intractable. We also prove lemmas that will be used throughout.

The formalism for sampling problems, which goes back to at least [63], begins with a finite alphabet Σ and a binary relation between words in this alphabet $R \subseteq \Sigma^* \times \Sigma^*$. We interpret $(x, y) \in R$ as asserting that y is a solution to the instance x. For example, we can define a binary relation R as those (x, y) such that x encodes a graph $G(x)$ and y encodes the edges of a simple cycle of $G(x)$. We will consider only those relations that can be verified efficiently, which are called p-relations:

Definition 2.1 (p-relations, [63]). *A relation $R \subseteq \Sigma^* \times \Sigma^*$ is a p-relation if there is a deterministic polynomial time Turing machine that recognizes $R \subseteq \Sigma^* \times \Sigma^*$ and if there is a polynomial p such that $\forall x$, $(x, y) \in R$ implies that $|y| \leq p(|x|)$. We define $R(x) = \{y \in \Sigma^* : (x, y) \in R\}$.*

Now we define the sampling problems we will be considering:

Definition 2.2 (Family of p-distributions). *A family of p-distributions is defined by a p-relation R and function $f : \Sigma^* \to \mathbb{Q}_{\geq 0}$. For each instance $x \in \Sigma^*$ with $R(x) \neq \emptyset$, we require that f is not identically zero on $R(x)$. For such an instance x, we associate a probability distribution p_x on $R(x)$, where $y \in R(x)$ has weight proportional to $f(y)$. The uniform distribution on R is defined by taking f to be identically 1.*

Definition 2.3 (Sampling problem). *To each family of p-distributions (R, p_x), there is an associated sampling problem, which we also refer to as (R, p_x):*

$P = (R, p_x)$ Sampling
Input: $x \in \{x \in \Sigma^* : R(x) \neq \emptyset\}$
Output: A sample drawn according to p_x.

Similar to approximation algorithms in the deterministic case, we can ask if Turing machine "almost" solves a sampling problem:

Definition 2.4 (α-almost solving a sampling problem). *Suppose that $P = (R, p_X)$ is some sampling problem. Let $\alpha \in [0, 1]$. We say that a probabilistic Turing machine M* α-almost solves *$P = (R, p_x)$* Sampling *if for all instances X with $R(X) \neq \emptyset$, $M(X)$ accepts X at least half the time and then outputs a sample from a distribution q_X^M, where $||q_X^M - p_X||_{TV} \leq \alpha$. In the case $\alpha = 0$, we say that M solves the sampling problem.*

We will use the complexity class RP to describe the intractability of a sampling problem.

DEFINITION 2.5 (The class RP [15]). RP *is the class of languages* $L \subseteq \Sigma^*$ *such that there is a polynomial time probabilistic Turing machine* M *and a constant* $\epsilon > 0$ *so that, if* $x \notin L$, $M(x)$ *always rejects, and if* $x \in L$, $M(x)$ *accepts with probability at least* ϵ.

It is widely believed that RP $\neq$ NP; this belief follows from the widely believed conjectures that NP $\neq$ P [10] and BPP = RP = P [58]. Based on this reasoning, and arguments in the style of [63, Proposition 5.1] or [85, Theorem 1.17], which argue that there is likely no efficient algorithm for a sampling problem by showing that the existence of an efficient sampler would imply RP = NP, we make the following definition for when a sampling problem is intractable:

DEFINITION 2.6 (Intractability of a sampling problem). *We say that a sampling problem* P *is* intractable *on a language (or class) of instances* $\mathscr{C}$ *if for all* $\alpha < 1$, *the existence of a polynomial time probabilistic Turing machine that* α*-almost uniformly samples from* P *for all instances in* $\mathscr{C}$ *implies that* RP = NP.

Lemma 2.8 below abstracts the repetitive part of most proofs showing that a sampling problem is intractable. To state it cleanly, we make the following definition, which takes a p-relation Q, contained inside a p-relation S, and describes the set of instances that have solutions:

DEFINITION 2.7 (Decision problem on a p-relation). *Let* S *be a* p*-relation. A* decision problem *in* S *is a* p*-relation* Q, *such that* $Q \subseteq S$, *with an associated language* $L_Q = \{x \in \Sigma^* : Q(x) \neq \emptyset\}$. *If* $\mathscr{C} \subseteq \Sigma^*$ *is a language, and* $L_Q(\mathscr{C}) := \{x \in \mathscr{C} : Q(x) \neq \emptyset\}$ *is* NP*-complete, then we will say that* Q *is a decision problem in the* p*-relation* S *which is* NP*-complete on the language* $\mathscr{C}$.

LEMMA 2.8 (Lucky guess lemma). *Consider some sampling problem* $P = (R, p_x)$. *Let* Q *be some decision problem in a* p*-relation* S, *which is* NP*-complete on the language* $\mathscr{C}$. *Suppose the following assumptions hold for some polynomials* $p_m(n)$, $m \in \mathbb{N}_{\geq 1}$:

- *There is a* p_m*-time Turing machine* B_m *such that for any instance* $x \in \mathscr{C}$, B_m *constructs some* $B_m(x) \in \Sigma^*$ *with* $R(B_m(x)) \neq \emptyset$.
- *There is another* p_m*-time Turing machine* M_m *that computes a map* $\pi_m : R(B_m(x)) \to S(x)$.
- *(Probability Concentration) If* $|Q(x)| \geq 1$ *and if* C *is a random variable distributed according to* P *on* $R(B_m(x))$, *then* $\mathbb{P}(\pi_m(C) \in Q(x)) \geq 1 - 1/m$.

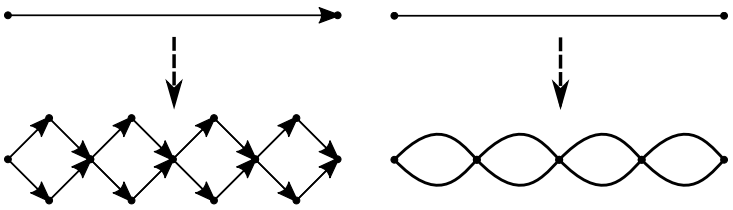
Then, P *is intractable on the language* $B(\mathscr{C}) = \{B(x) : x \in \mathscr{C}\}$.

Proof. Fix $\alpha < 1$, and take $m = \lceil \frac{2}{1-\alpha} \rceil$. We fix $B = B_m, M = M_m$. Assume that there exists a polynomial time probabilistic Turing machine G that α-almost solves P on $B(\mathscr{C})$. We claim that Algorithm 2.1 gives an RP-algorithm for $L_Q(\mathscr{C})$. Algorithm 2.1 runs in polynomial time, since for any $X \in \mathscr{C}$, constructing $B(X)$, sampling C with G and computing $\pi(C)$ with M takes time polynomial in $|X|$. Thus, we only have to prove that the algorithm succeeds with the correct error bounds. Since Algorithm 2.1 clearly has no false positives, we only need to check that there is a constant lower bound on the true positive rate. We will show that if $|Q(X)| \geq 1$, then the probability of success is at least $1/m$. Suppose that q_Y is the distribution over $R(Y)$ of outputs of G on input Y. Suppose that $A = \{C \in R(B(X)) : \pi(C) \in Q(X)\}$. Since $\|p_{B(X)} - q_{B(X)}\|_{TV} < \alpha$, and in particular $p_{B(X)}(A) - q_{B(X)}(A) < \alpha$, it follows that $q_{B(X)}(A) > p_{B(X)}(A) - \alpha \geq 1 - 1/m - \alpha \geq 1/m$. Hence, with probability at least $1/m$, the sample drawn by G from $R(B(X))$ will land in A. In other words, if $|Q(X)| \geq 1$, then Algorithm 2.1 will answer YES with probability at least $1/m$. Since $L_Q(\mathscr{C})$ is NP-complete, it would follow that NP = RP. Since this argument holds for all $\alpha < 1$, P is intractable on $B(\mathscr{C})$. ∎

Algorithm 2.1 Lucky Guess

Input: G, M, B and $x \in \mathcal{C}$ as in the proof of Lemma 2.8.

1: Construct $B(x)$
2: Let C be the output of G on $B(x)$
3: **if** $M(C) \in Q(x)$ **then**
4: return `YES`
5: **else**
6: Return `NO`.



(a) Directed (b) Undirected

Figure 2.1: The directed version from [63] and its undirected bigons counter part.

Certain calculations appear repeatedly when checking the probability concentration hypothesis of Lemma 2.8. We isolate them here:

LEMMA 2.9. *If $H, N \geq 0$ and $H \geq DN$ for some $D > 0$, then $\frac{H}{H+N} \geq \frac{D}{1+D}$.*

LEMMA 2.10. *Fix $q \geq 2$. Then for any $e \in \mathbb{N}$ and $S \geq 1$ if $d \geq 2$ and $d \geq 4(\frac{\log_2(S)+e}{\log_2(q)})^2$ then $q^d \geq Sd^e$.*

Proof. It suffices to pick d so that $\frac{d}{\log(d)} \geq \frac{\log(S)+e}{\log(q)}$. Since $\frac{d}{\log_2(d)} \geq \frac{1}{2}\sqrt{d}$ for $d \geq 2$, the claim follows.

2.2. Intractability of Uniformly Sampling Simple Cycles. The seminal paper [63] proves that the following sampling problem is intractable:

GENDIRECTEDCYCLE
Input: A directed graph G.
Output: An element of the set of directed simple cycles of G, selected uniformly at random.

THEOREM 2.11 ([63], Proposition 5.1). *The sampling problem* GENDIRECTEDCYCLE *is intractable on the class of directed graphs.*[1]

We refer the reader to the original paper for the proof. The idea is to concentrate probability on longer cycles by replacing edges with chains of diamonds, as in Figure 2.1(a), which has the effect of increasing the number of ways to traverse a cycle by a quantity which grows at a exponential rate proportional to the length of the cycle.

Similarly, one can consider an undirected, plane graph version of the same problem:

GENSIMPLECYCLE
Input: An undirected, plane graph G.
Output: An element of $SC(G)$, selected uniformly at random.

Although similar to Theorem 2.11, will include the proof that GENSIMPLECYCLE is intractable on the class of plane graphs as a preface to our other results. To do so, we formalize an analog of the chain of diamonds construction:

DEFINITION 2.12 (Chain of bigons, projection map). *Let G be an (undirected) graph. Let $B_d(G)$ denote the graph obtained from G by replacing each edge by a* chain of d bigons. *Formally, we subdivide each edge into d edges and then add a parallel edge for every edge. This construction is illustrated in Figure 2.1(b). There is a natural map $\pi_d : SC(B_d(G)) \to 2^{E(G)}$, which collapses the chains of bigons. Formally, $\pi_d(C) = \{e \in E(G) : B_d(e) \cap C \neq \emptyset\}$.*[2]

[1] We want to acknowledge a helpful Stack Exchange conversation [2] with Heng Guo that alerted us to this theorem.

[2] For there to be a Turing machine M as in Lemma 2.8 that simulates π_d, the collection of $B_d(e)$ for $e \in E(G)$ has to be part of the encoding of the graph $B_d(G)$ as a string of Σ^*, and B must construct an encoding of $B_d(G)$ with this information. We assume that all of this is true, and make sense of $SC(B_d(G))$ and similar objects in terms of the underlying graph. In general we will omit discussion of this level of detail.

PROPOSITION 2.13 (Adaptation of [63], Proposition 5.1). GENSIMPLECYCLE *is intractable on the class of simple plane graphs.*

Proof. Given a polynomial time probabilistic Turing machine M which α-solves GENSIMPLECYCLE on the class of simple plane graphs, we obtain one that α-almost solves GENSIMPLECYCLE on the class of plane graphs by subdividing each edge of a given graph. We thus let $\mathscr{C}$ be the class of plane graphs, and we will now work on checking that the conditions for Lemma 2.8 can be satisfied. For $G \in \mathscr{C}$, with $n = |V(G)$, we fix m and take $d = n^2 + m$. Observe that $\mathrm{Im}(\pi_d) = SC(G) \cup E(G)$. If $X \in SC(G)$ is a simple cycle with m edges, then $|\pi_d^{-1}(X)| = 2^{dm}$. For $e \in E(G)$, $|\pi_d^{-1}(e)| = d$. Thus, if $\mathscr{H} \subset SC(G)$ is the set of Hamiltonian cycles of G, and $|\mathscr{H}| \neq 0$, then $|\pi_d^{-1}(\mathscr{H})| \geq 2^{dn} \geq 2^m 2^{n^2} 2^{d(n-1)} \geq 2^m |\pi_d^{-1}(2^{E(G)} \setminus \mathscr{H})|$; here we have used 2^{n^2} as a crude upper bound on $|SC(G) \cup E(G)|$. Thus, by Lemma 2.9, $\frac{|\pi_d^{-1}(\mathscr{H})|}{|SC(B_d(G))|} \geq \frac{2^m}{1+2^m} \geq 1 - 1/m$. Define polynomial-time Turing machines B and M so that $B(G) = B_d(G)$ and $M(C) = \pi_d(C)$, and set $S(G) = E(G) \cup SC(G)$, $Q(G) =$ HamiltonianCycles(G) and $R(G) = SC(G)$ for all $G \in \mathscr{C}$. Since the Hamiltonian cycle problem is NP-complete on $\mathscr{C}$ [49], the conditions of Lemma 2.8 are satisfied.

2.3. Intractability of Sampling from $P_2(G)$. As we discussed in §1.3, we are interested in the problem of uniformly sampling connected 2-partitions.

GENCONNECTED2PARTITION
Input: A graph G.
Output: An element of $P_2(G)$, selected uniformly at random.

We recall a fact about plane duality, which will connect Proposition 2.13 above to GENCONNECTED2PARTITION.

THEOREM 2.14 ([47]). *Let G be a plane graph and G^* its plane dual. Then, there is an polynomial time computable bijection between $SC(G^*)$ and $\mathscr{P}_2(G)$.*[3]

THEOREM 2.15. GENCONNECTED2PARTITION *is intractable on the class of plane graphs.*

Proof. If G is a plane graph, then a polynomial algorithm to sample uniformly $P_2(G^*)$ gives an algorithm to sample uniformly from $SC(G)$ by Theorem 2.14. Now the result follows from Proposition 2.13. ∎

This theorem implies that the broadest version of uniform sampling from the space of graph partitions is intractable. While already this observation is notable given how little is known about the graphs that appear in redistricting, our discussion does not end here. Rather, §2.3 will be strengthened in Theorem 2.17 and Theorem 2.45. In §3, we will also highlight how the probability concentration gadgets identify concrete issues with Markov chains used for sampling partitions.

2.4. Intractability of Uniformly Sampling Balanced Partitions. For applications in redistricting, the blocks of a connected partition should be roughly equal in population. This motivates studying the problem of sampling *balanced* partitions:

DEFINITION 2.16 (ϵ-balanced simple cycles and 2-partitions). *Let $SC^\epsilon(G)$ be the set of ϵ-balanced simple cycles of a plane graph G, such that if $\{A, B\}$ is the dual connected 2-partition of G^* then $1 - \epsilon \leq \frac{|A|}{|B|} \leq 1 + \epsilon$ and $1 - \epsilon \leq \frac{|B|}{|A|} \leq 1 + \epsilon$. Similarly, we say that a partition $(A, B) \in P_2(G)$ is ϵ-balanced if these inequalities hold for $\{A, B\}$; we define $P_2^\epsilon(G)$ to be the set of such partitions.*

ϵ-BALANCED UNIFORM 2-PARTITION SAMPLING
Input: A plane graph G.
Output: An element of $P_2^\epsilon(G)$ selected uniformly at random.

The existence of a 0-*balanced* 2-partition in a graph G is not obvious. In fact, determining if there exists a balanced connected 2-partition of a given graph G is NP-complete:

[3] We wish to acknowledge a helpful Stack Exchange discussion with Mikhail Rudoy which first drew our attention to this theorem [6].

THEOREM 2.17 ([45], Theorem 2.2[4]). *The decision problem of whether a given connected graph G has a 0-balanced, connected 2-partition is* NP*-complete.*

Given Theorem 2.17, the problem of uniformly sampling from the set of 0-balanced connected 2-partitions is vacuously intractable. To circumvent this issue, we focus on the case where G is 2-connected, since in that case a 0-balanced 2-partition always exists and can be constructed in polynomial time by constructive versions [90] of the Győri–Lovász Theorem [53, 72] for 2-partitions. As in the previous section, our strategy is to work with simple cycles, rather than connected partitions. In particular, the following classical theorem will let us translate a statement about Hamiltonian cycles into a statement about balanced Hamiltonian cycles:

THEOREM 2.18 (Grinberg's Theorem, [51]). *Let G be a plane graph. Assign to each face F a weight $deg(F) - 2$, where $deg(F)$ is the number of edges in F. If H is a Hamiltonian cycle of G, then the total weight of the faces inside of H is equal to the total weight of the faces outside of H.*

Grinberg's Theorem implies that every Hamiltonian cycle of a maximal plane graph is balanced, since every face is a triangle. Since the problem of determining whether a maximal plane graph has a Hamiltonian cycle is NP-complete [93], it follows that the problem of determining whether a maximal plane graph has a *balanced* Hamiltonian cycle is NP-complete.[5]

We begin by defining the map π and proving the probability concentration result used to apply Lemma 2.8. In particular, recalling the definition of π_d (Definition 2.12), we define $\pi_d^\epsilon : SC^\epsilon(B_d(G)) \to 2^{E(G)}$ as the restriction of $\pi_d : SC(B_d(G)) \to 2^{E(G)}$ to the ϵ-balanced simple cycles of G. We derive the necessary inequalities in the following lemma:

LEMMA 2.19. *Let $G = (V, E)$ be a maximal plane graph, with $n = |V(G)|$. Let C be a Hamiltonian cycle of G, and let A be any non-Hamiltonian cycle of G. Let $\epsilon \geq 0$. Then we have*

$$|(\pi_{2d}^\epsilon)^{-1}(C)| \geq |(\pi_{2d}^0)^{-1}(C)| = \binom{2dn}{dn} \geq \frac{2^{2dn}}{2dn+1}, \text{ and} \tag{2.1}$$

$$|(\pi_{2d}^\epsilon)^{-1}(A)| \leq |(\pi_{2d}^\infty)^{-1}(A)| \leq 2^{2d(n-1)}. \tag{2.2}$$

If $\mathscr{H}$ is the set of Hamiltonian cycles of G, and $C \in \mathscr{H}$, then for $d \geq 4\lceil(m + n^2/2 + 3/2 + \log(n))^2\rceil$,

$$|(\pi_{2d}^\epsilon)(\mathscr{H})| \geq 2^{2m}|\pi_{2d}^{-1}(2^{E(G)} \setminus \mathscr{H})|. \tag{2.3}$$

Proof of Equation (2.1) and Equation (2.2). Since every Hamiltonian cycle of G is balanced, the only way to lift the cycle to a balanced simple cycle of $B_{2d}(G)$ is to take the inward edge along exactly half of the bigons. For the lifts of non-Hamiltonian cycles, we can bound the number of lifts to a balanced simple cycle by the number of lifts to a cycle. These inequalities then follow from the standard fact that $\frac{2^{2r}}{2r+1} \leq \binom{2r}{r}$

Proof of Equation (2.3). By Lemma 2.10, for $d \geq 4\lceil(m + n^2/2 + 3/2 + \log(n))^2\rceil$ we have

$$2^{2dn} \geq (2^{2m+n^2+2}n)d2^{2d(n-1)} \geq 2^{2m}2^{n^2}(2dn+1)2^{2d(n-1)},$$

and thus

$$|(\pi_{2d}^\epsilon)(\mathscr{H})| \geq |(\pi_{2d}^\epsilon)(C)| \geq \frac{2^{2dn}}{2dn+1} \geq 2^{2m}2^{n^2}2^{2d(n-1)} \geq 2^{2m}|\pi_{2d}^{-1}(2^{E(G)} \setminus \mathscr{H})|.$$

PROPOSITION 2.20. *Fix $\epsilon \geq 0$. Then,* ϵ-BALANCED UNIFORM SIMPLE CYCLE SAMPLING *is intractable on the class of graphs of the form $B_d(G)$, where $d \geq 1$ and G is any maximal plane graph.*

Proof. To prove this, we fit what we have calculated into the format of Lemma 2.8. Fix m. Then we take $d = 4\lceil(m + n^2/2 + 3/2 + \log(n))^2\rceil$, which is polynomial in G, and set $B(G) = B_{2d}(G)$, and M to

[4]The phrase "k-partition" in [45] has a different meaning from the way we use it here. In their notation, a k-partition is a connected partition where each piece has size k. What we call a balanced connected 2-partition, they call an $n/2$ partition. The theorem is stated here in our notation.

[5]The authors wish to thank Gamow from Stack Exchange for a helpful comment [7] and directing us towards [93].

compute π_{2d}. The probability concentration hypothesis follows from Lemma 2.9 and Equation (2.3), since they show that $\frac{|(\pi^{\epsilon}_{2d})^{-1}(\mathscr{H})|}{|SC^{\epsilon}(G)|} \geq \frac{4^m}{1+4^m} \geq 1 - 1/m$. Finally, we set Q to be the Hamiltonian cycles, and $\mathscr{C}$ to be the class of maximal plane graphs.

THEOREM 2.21. *Fix $\epsilon \geq 0$. Then ϵ-*BALANCED UNIFORM 2-PARTITION SAMPLING *is intractable on the class of 2-connected plane graphs.*

Proof. $(B_d(G))^*$ is 2-connected when G is a maximal plane graph. The claim now follows by Proposition 2.20 using Theorem 2.14. ∎

REMARK 2.22. *These arguments work for any reasonable definition of nearly balanced that considers partitions with $|A| = |B|$ to be balanced.*

2.5. Sampling intractability on maximal plane Graphs with bounded degree. In this section, we improve the results from §2.3 by showing that the connected 2-partition sampling problem is intractable on the class of maximal plane graphs with vertex degree bounded by 531. This will shrink the gap between our theoretical intractability statements and the graphs used to study redistricting. To obtain our result, we will start in §2.5.1 by proving a corresponding NP-completeness theorem, building on the results in [49]. Second, in §2.5.2, we will describe a construction for concentrating probability on the longer simple cycles by providing a simple cycle with many paths through a vertex, rather than with many paths through an edge. Finally, in §2.5.3 we will show how to tie the intractability argument together. We will reuse the gadgets from this section in Corollary 3.18 to construct explicit counterexamples to heuristic sampling algorithms.

2.5.1. Hamiltonian cycle is NP-complete on cubic, 3-connected plane graphs with face degree ≤ 177.

DEFINITION 2.23 ($3CCP$ graphs with bounded face degree). *Recall that a $3CCP$ graph is one that is 3-connected, cubic, simple, and plane. Let $\mathscr{C}_m$ denote the collection of $3CCP$ graphs with face degree $\leq m$. Let $\mathscr{C} = \bigcup_{k\geq 3} \mathscr{C}_k$.*

DEFINITION 2.24 (Hamiltonian cycle problem). *If $\mathscr{D}$ is a language of graphs, let $\mathscr{D}$-$HAM = \{G \in \mathscr{D} : G$ is Hamiltonian$\}$*

Our goal in this section is to prove the following theorem:[6]

THEOREM 2.25. *$\mathscr{C}_{177}$-HAM is* NP*-complete.*

We will prove a reduction from the following theorem:

THEOREM 2.26 ([49]). *$\mathscr{C}$-HAM is* NP*-complete*

To obtain Theorem 2.25 from Theorem 2.26, we will show that we can take $G \in \mathscr{C}$ and construct a $G' \in \mathscr{C}_{177}$ such that G' has Hamiltonian cycle if and only if G does. We will obtain G' from G by using an algorithm that subdivides the large faces repeatedly (Algorithm 2.2) in polynomial time (Proposition 2.33). The gadget that we use to subdivide large faces comes from the proof of Theorem 2.26 in [49], in particular, we use their 3-way OR gate gadget. We now review some relevant properties of that 3-way OR (3OR) that will be used in the reduction:

DEFINITION 2.27 ($3OR$ Gadget, [49]). *The $3OR$ gadget is pictured in Figure 2.2. This gadget has three distinguished sets of attaching nodes, each of which consists of a path graph with 6 nodes.*

DEFINITION 2.28 ($3OR$ insertion). *A $3OR$ gadget can be inserted into a face F of a plane graph by picking 3 edges $\{e_1, e_2, e_3\}$ of F, replacing each edge e_i with a path containing 6 nodes, and gluing each of the distinguished segments of the $3OR$ to one of those subdivided edges. For the plane embedding, we place the rest of the $3OR$ into the interior of F, as in Figure 2.3. We will refer to this operation as* inserting a *$3OR$ into F at the edges $\{e_1, e_2, e_3\}$.*

[6] The authors are grateful to Pálvölgyi Dömötör Honlapja for suggesting the proof strategy used in this section [3]. We are of course responsible for all errors in our execution of the strategy.

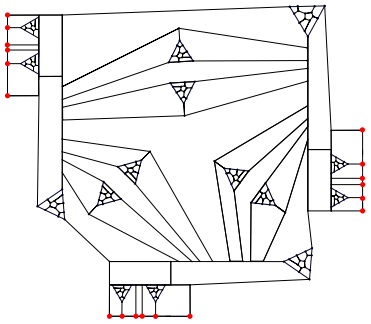


Figure 2.2: The 3OR gadget. See Figure A.1 in the appendix for more detail. The distinguished attaching nodes are colored red.

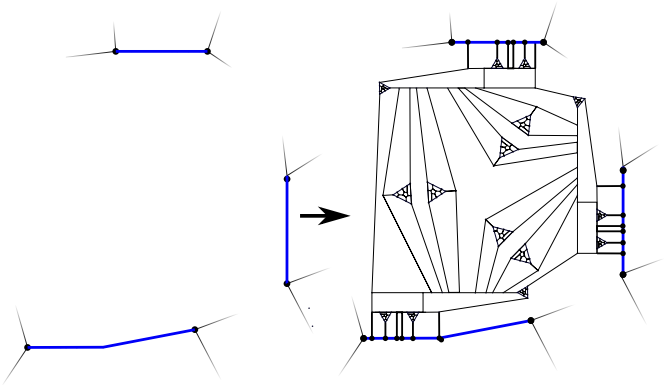


Figure 2.3: Inserting a $3OR$. See Figure A.1 for more detail.

LEMMA 2.29. *Let H be a $3CCP$ graph, F a face of H, and $\{e_1, e_2, e_3\}$ edges of H. Construct a graph H' by attaching a $3OR$ gadget to the edges $\{e_1, e_2, e_3\}$. Then H' has a Hamiltonian cycle if and only if H has a Hamiltonian cycle containing at least one of the e_i. Additionally, H' is a $3CCP$ graph.*

Proof [49]. This amounts to an analysis of the local states, which are described in [49, Fig. 6B]. The proof that these are the only possible local states exploits detailed properties of the $3OR$ gadget; the reader can consult [49] for details. The proof that H' is 3-connected follows from Lemma A.1, and checking that H' is cubic and planar is straightforward. □

Our strategy will be to take large faces and subdivide them using $3OR$ gadgets:

DEFINITION 2.30 (Subdivision). *Let F be a face of H. A* subdivision of F *is a graph H' obtained by taking 3 edges of F, say e_1, e_2, e_3, where e_1 and e_2* share a vertex*, and inserting a $3OR$ into F at e_1, e_2, e_3. See Figure 2.3 for an illustration of this definition. In Figure A.1 we label a few regions of this subdivision for reference:* 3 adjacent faces*, a* pocket*, and* 2 large faces*.*

The difference between 'subdivision' and 'inserting a $3OR$' is that we require that 2 of the edges used are adjacent. The following proposition shows that we can subdivide faces without changing Hamiltonicity:

PROPOSITION 2.31. *Let H be any $3CCP$ graph, and let F be a face of H. Let v be a vertex of F, let e_1 and e_2 be the edges of F adjacent to v, and let e_3 be any other edge of F. Let H' be the graph obtained by subdividing H at e_1, e_2 and e_3. Then, H has a Hamiltonian cycle if and only if H' has a Hamiltonian cycle. See Figure A.1.*

Algorithm 2.2 Subdivision of faces with more than d edges

Input: A plane graph H and $d \in \mathbb{N}$.

1: Let F be any face of H with maximum degree
2: **if** $deg(F) \leq d$ **then**
3: terminate and return H
4: **else**
5: Use a $3OR$ gadget to optimally subdivide F (Definition 2.32). Set this subdivided graph as H.
6: Return to 1.

Proof. Since H is cubic, every Hamiltonian cycle of H uses all but one of the edges at each vertex. Thus, any Hamiltonian cycle uses at least one of $\{e_1, e_2\}$, allowing us to apply Lemma 2.29.

Since our goal is to reduce the face degree, we define the subdivisions that optimally decrease degree:

DEFINITION 2.32 (Optimal subdivision). *Let F be a face of a $3CCP$ graph G. An* optimal subdivision *of F is any subdivision of F that minimizes the maximum degree of the two large faces (cf. Definition 2.30) of the subdivision.*

Algorithm 2.2 takes a graph H and a parameter d, and—if it terminates—returns a graph that has no faces of degree $> d$. We will use this algorithm to reduce to an instance of $\mathscr{C}_{177}$-HAM from one of $\mathscr{C}$-HAM, so we must show that Algorithm 2.2 terminates in polynomial time for an appropriate choice of d. To determine the necessary d, we need the following lemma:

LEMMA 2.1. *Let G be a $3CCP$ graph. Suppose that F is a face of degree f. Suppose we make an optimal subdivision of F at edges e_1, e_2, e_3, where e_1 is adjacent to e_2. Let F_i be the face in G adjacent to e_i for $i = 1, 2, 3$. Then, the following hold:*

- *Each F_i is distinct. Moreover, the degree of each F_i increases by 6.*
- *The two large faces (Figure A.1) inside of what was originally F each have degree $\lfloor f/2 \rfloor + 10$ and $\lceil f/2 \rceil + 10$*
- *The gadget itself introduces 33 faces of degree 4, 75 faces of degree 5, 9 faces of degree 7, 15 faces of degree 8, 4 faces of degree 9, 3 faces of degree 10, 3 faces of degree 12, and 3 of degree 14. We call these the "small faces."*
- *A face of degree 10 is introduced, which is labelled "the pocket" in Figure A.1.*

Proof. Figure A.1 in the appendix can be used to count the degrees of these faces, and the number of the small faces of different sizes. The fact that each F_i is distinct follows from the fact that the graph is $3CCP$. ∎

PROPOSITION 2.33. *Let H be a $3CCP$ graph with n nodes. As long as $d \geq 178$, Algorithm 2.2 terminates in time polynomial in $|H|$.*

Proof. We will consider a single step in the subdivision algorithm and show that in each step a certain nonnegative energy function decreases by at least one. Since the energy function starts off with value $O(n^2)$, the proposition will follow.

Let f_j, $j = 1, \ldots, |F(H)|$, and f'_k, $k = 1, \ldots, |F(H')|$ refer to the face degrees in some enumerations of the faces of H, before and after one step of Algorithm 2.2, respectively. Assume that f_1 corresponds to the face F_1 being subdivided during that step and that f_2, f_3, f_4 correspond to the faces adjacent to F_1 along the edges where the $3OR$ gadget is being added. Let $S = \sum f_i^2$ and $S' = \sum f'^2_k$. By Lemma 2.1 and notating the degrees of all the small faces by c_i, we have that $S' = S - f_1^2 + \sum c_i^2 + 10^2 + (\lfloor f_1/2 \rfloor + 10)^2 + (\lceil f_1/2 \rceil + 10)^2 + \sum_{j \in \{2,3,4\}} ((f_j + 6)^2 - f_j^2)$.

Thus, if d is a positive integer so that $-d^2 + \sum c_i^2 + 10^2 + (\lfloor d/2 \rfloor + 10)^2 + (\lceil d/2 \rceil + 10)^2 + 3((d+6)^2 - d^2) \leq -1$, then whenever there is a face of degree $> d$ one step of the subdivision algorithm reduces the energy S by at least one. The precise computation of the smallest such d depends on the counts of the c_i listed in Lemma 2.1. Using these counts, it can be checked that taking $d = 178$ suffices to ensure that the energy decreases by at least one in each step.

Finally, note that the initial energy S is bounded by $(\sum_i f_i)^2 = (2|E(H)|)^2 = O(|V(H)|^4)$. Therefore, after $O(|V(H)|^4)$ subdivision steps, Algorithm 2.2 with $d = 178$ terminates. Since each step of Algorithm 2.2

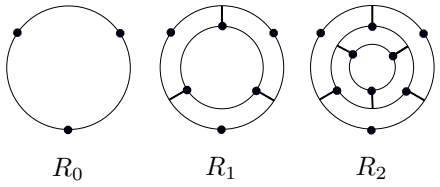


Figure 2.4: The first 3 terms in the sequence of gadgets.

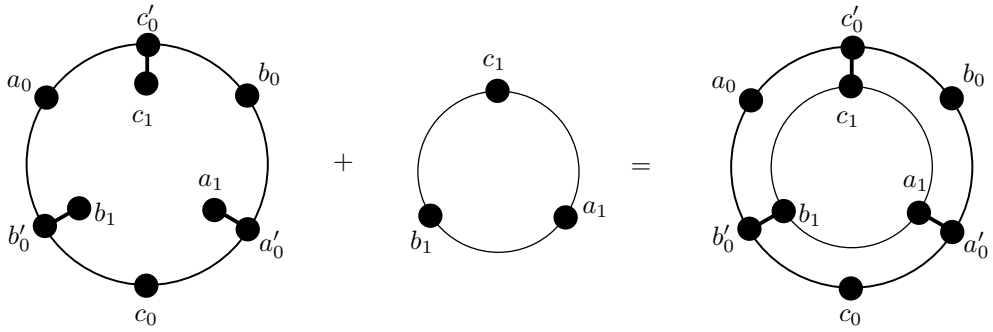


Figure 2.5: The node labels in the recursive construction

takes polynomial time, the result follows. ∎

In particular, we can use Algorithm 2.2 to eliminate all of the faces of degree greater than or equal to 178. Combining this with Theorem 2.26, we can now prove Theorem 2.25:

Proof of Theorem 2.25. Let $H \in \mathscr{C}$. Apply Algorithm 2.2 to obtain an $H' \in \mathscr{C}_{177}$, such that H' has a Hamiltonian cycle if and only if H has one. Constructing H' takes polynomial time, by Proposition 2.33. ∎

2.5.2. The Node Replacing Gadgets R_d. In this section, we will construct the corresponding gadget to the chain of bigons, which will allow us to concentrate probability on the longer cycles while remaining $3CCP$. Instead of replacing edges with a chain of d bigons, which allowed for 2^d choices of ways to route through that edge, the gadgets R_d we construct here will replace cubic vertices and allow for $\Theta(5^d)$ choices through that vertex.[7] The first few R_d's are displayed in Figure 2.4, and we give a definition below:

Definition 2.34 (The probability concentration gadgets R_d, C_d, R_d'). *Define R_0 to be a 3-cycle, with nodes labeled with (a_0, b_0, c_0). For each $d \geq 0$, we will construct R_{d+1} from R_d. First, we construct R_d' from R_d by subdividing the edges $\{x_d, y_d\}$ for all $x \neq y \in \{a, b, c\}$. The node that subdivided the edge $\{x_d, y_d\}$ gets labelled z_d', where $\{x, y, z\} = \{a, b, c\}$. For each $x \in \{a, b, c\}$, we attach a node x_{d+1} and an edge $\{x_d', x_{d+1}\}$. Then, we separately build a 3 cycle C_{d+1} with nodes labelled by $(a_{d+1}, b_{d+1}, c_{d+1})$. We obtain R_{d+1} by gluing R_d' to C_d by identifying the nodes with the same labels. See Figure 2.5 for an illustration of this construction.*

To show probability concentration, we will need to compute the number of choices a simple cycle will have when passing through an R_d gadget, as well as the number of simple cycles internal to an R_d gadget. The latter we know how to describe as $|SC(R_d)|$, and the former is captured by the following definition:

Definition 2.35 (Simple boundary links). *We call the simple paths in R_d that go from any two points $x \neq y \in \{a_0, b_0, c_0\}$* simple boundary links*, denoted $SBL(R_d; x, y)$, where $\{a_0, b_0, c_0\}$ are as in Definition 2.34. We denote $SBL(R_d) := SBL(R_d; a_0, b_0)$.*

Because of the rotational symmetry of the gadget, $|SBL(R)|$ does not change if we choose a different two element subset of $\{a_0, b_0, c_0\}$ as the start and stop vertices. We next introduce notation that will be

[7] The inspiration for this construction came from [93], wherein one step a reduction from Hamiltonian cycle on $3CCP$ graphs to Hamiltonian cycle on maximal plane graphs is to replace cubic vertices by a certain gadget.

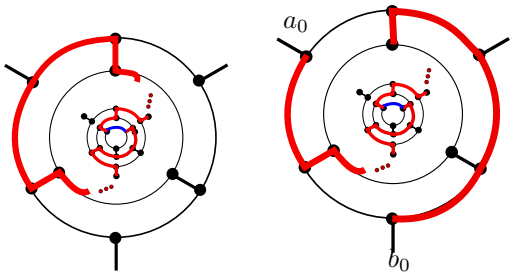


Figure 2.6: The decompositions used for Equation (2.4) and Equation (2.6). The S_d (resp. D_d) part is in red. Observe that there are two options for the simple path $R_d[C_d]$, one of which is colored blue.

useful when computing $|SC(R_n)|$ and $|SBL(R_n)|$:

DEFINITION 2.36 (Simple paths from X to Y). *For any graph G, with $X, Y \subseteq V(G)$, we let $SP_{X,Y}(G)$ denote the set of simple paths in G that start in X, and stop at the first positive time they reach Y:*

$$SP_{X,Y}(G) = \{\gamma = (x_0, x_1, \ldots, x_n) : x_i \in V(G), \{x_i, x_{i+1}\} \in E(G), x_0 \in X, x_n \in Y, x_i \notin Y \text{ for } 0 < i < n\}.$$

THEOREM 2.37. *Let R_d be as in Definition 2.34. Then:*

$$|SC(R_d)| = \frac{1}{4}(3 \cdot 5^{d+1} - 8d - 11) \tag{2.4}$$

$$5^d \leq |SC(R_d)| \leq 5^{d+1} \tag{2.5}$$

$$|SBL(R_d)| = \frac{1}{2}(5^{d+1} - 1) \tag{2.6}$$

$$5^d \leq |SBL(R_d)| \leq 5^{d+1}. \tag{2.7}$$

Proof of Equation (2.4). We partition the simple cycles in R_d into those that touch C_d and those that do not. Those simple cycles that do not touch C_d can be identified with simple cycles in R_{d-1}. To describe the simple cycles that touch C_d, we start by defining $S_d = \{X \in SP_{V(C_d),V(C_d)}(R_d) : X \cap E(C_d) = \emptyset\}$. Among the cycles that touch C_d, there is C_d itself, and there are the cycles that can be decomposed into an element of $SP_{V(C_d),V(C_d)}(R_d[C_d])$ along with an element of S_d, as in Figure 2.6. Thus, $SC(R_d) = SC_{d-1} + 1 + 2S_d$.

It can be checked that $S_d = 5S_{d-1} + 6$, by analyzing the number of ways to extend an element of S_{d-1} to an element of S_d and by accounting for the six elements of S_d not obtained by extensions of S_{d-1}, as in Figure 2.7. This second calculation also shows that $S_1 = 6$. We can solve this recurrence relation to conclude that $S_d = (3/2)(5^d - 1)$. Hence, from $SC_d = SC_{d-1} + 1 + 2S_d$ we have that $SC_d = SC_{d-1} + 1 + 3(5^d - 1)$. As $SC_0 = 1$, we conclude that $SC_d = 1/4(3 \cdot 5^{1+d} - 8d - 11)$.

Proof of Equation (2.6). We partition the simple boundary links in R_d into those that pass through C_d and those that do not. Those elements of $SBL(R_d)$ that do not touch C_d can be identified with $SBL(R_{d-1})$. To analyze those that pass through C_d, we define D_d to be the set of pairs of disjoint simple paths, one from a_0 that stops at the first point it touches C_d, and the other from b_0 that stops at the first point it touches C_d. The elements of $SBL(R_d)$ that touch C_d can be decomposed into an element γ of D_d and one of the two simple paths in $R_d[C_d]$ that connect the points where γ meets C_d, as in Figure 2.6. Thus, $BL_{d+1} = BL_d + 2D_d$.

We can compute that $D_{d+1} = 5D_d$, by analyzing how elements of D_d can be extended to elements of D_{d+1}, as in Figure 2.7. As $D_0 = 1$, we have that $D_d = 5^d$. From $BL_{d+1} = BL_d + 2D_d$, we have that $BL_{d+1} = BL_d + 2(5^{d+1})$. As $BL_0 = 2$, we can solve the recurrence to find that $BL_d = (1/2)(5^{d+1} - 1)$. ∎

Proof of Equation (2.5) and Equation (2.7). These follow directly from Equations (2.4) and (2.6).

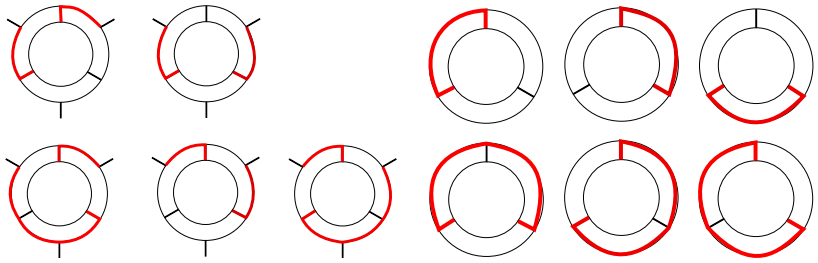


Figure 2.7: a) The five ways to extend an element of D_n to one of D_{n+1}, or one of S_n to one of S_{n+1}. b) The 6 elements of S_n that are not extensions of elements of S_{n-1}. The inner 3-cycle is C_n; most of R_n is not pictured.

2.5.3. Intractability on Maximal plane Graphs. We establish the analogous construction to replacing edges by chains of bigons and state a few results that we will need to finish the intractability proof.

Definition 2.38 ($G \to R_d(G)$ vertex replacement construction). *Given a cubic graph G, we let $R_d(G)$ denote the graph obtained by keeping the edges and replacing each vertex of G with a copy of R_d (Definition 2.34).*

Proposition 2.39. *The construction $G \mapsto R_d(G)$ sends $\mathscr{C}_m$ to $\mathscr{C}_{3m}$.*

See Appendix A.2 for proof of this proposition.

With the R_d construction in place, we analyze the relationship between $R_d(G)$ and G to show that R_d can be used to concentrate probability onto the longer simple cycles of G.

Definition 2.40 (Original edges, projection map). *There is a natural inclusion map $i : E(G) \to E(R_d(G))$. We will call the edges in the image of i the* original edges. *This lets us define a map $\pi_d : 2^{E(R_d(G))} \to 2^{E(G)}$ by $\pi_d(X) = i^{-1}(X)$.*

Lemma 2.41. *For any cubic graph G, $\mathrm{Im}(\pi_d) = SC(G) \cup \{\emptyset\}$ for $d \geq 0$.*

Proof. That $SC(H) \cup \{\emptyset\}$ is contained in the image is straightforward. Now, let $\beta \in SC(R_d(G))$. If $\pi(\beta)$ has degree 1 or 3 at a node $v \in V(G)$, then β had an odd degree node in $R_d(v)$, which impossible as β is a simple cycle. Moreover, $\pi(\beta)$ is connected. Since the simple cycles can be characterized as the non-empty connected edge subgraphs such that all nodes are degree 2, this concludes the proof. ∎

We now compute the probability concentration lemma necessary for applying Lemma 2.8:

Lemma 2.42. *Suppose that C is a Hamiltonian cycle of G, where G has $n \geq 2$ nodes. If $d \geq n^2 + n + m$, and X is a uniform sample from $SC(R_d(G))$, then $\mathbb{P}(\pi_d(X)$ is a Hamiltonian cycle of $G) \geq \frac{5^m}{1+5^m}$.*

Proof. Let H be the set of Hamiltonian cycles of G. From Equation (2.5) and Equation (2.7) it follows that $|\pi_d^{-1}(\emptyset \cup SC(G) \setminus H)| \leq 2^{n^2}5^{(d+1)(n-1)} + n5^{d+1} \leq 2^{n^2+1}5^{(d+1)(n-1)}$ and $|\pi_d^{-1}(H)| \geq 5^{dn}$. For $d \geq n^2 + n + m$, $5^{dn} \geq 5^m 2^{n^2+1} 5^{n-1} 5^{d(n-1)}$. The claim now follows by Lemma 2.9. ∎

Theorem 2.43. *If the Hamiltonian cycle problem is NP-complete on $\mathscr{C}_d$, then the problem of uniformly sampling simple cycles is intractable on the class of graphs $\mathscr{C}_{3d}$.*

Proof. We follow the notation of Lemma 2.8. For fixed m, we take $d = n^2 + n + m$. Define B by $B(G) = R_d(G)$, and M by the map π_d, and Q is the set of Hamiltonian Cycles of G. Lemma 2.42 assures that the conditions for Lemma 2.8 are satisfied. ∎

Corollary 2.44. *The $SC(G)$ uniform sampling problem is intractable on $\mathscr{C}_{531}$.*

Proof. Immediate from Theorem 2.25 and Theorem 2.43. ∎

We can now prove the main result of this section:

THEOREM 2.45. *The $P_2(G)$ uniform sampling problem is intractable on the class of maximal plane graphs with vertex degree ≤ 531.*

Proof. Since the dual graphs of those graphs in $\mathscr{C}_{531}$ are exactly the maximal plane graphs with vertex degree ≤ 531 and since simple cycles correspond bijectively to (unordered) connected 2-partitions under that duality, this result follows from Corollary 2.44. ∎

2.6. Intractability of uniformly sampling connected k-partitions. To obtain an intractability theorem about uniformly sampling connected k-partitions, we follow a similar approach as in previous sections. First, we recall a plane duality theorem, and then we prove that a relevant optimization problem is NP-complete. Finally, we introduce a gadget that concentrates samples on the certificates to that problem.

We remind the reader that $\mathscr{P}_k(G)$ denotes the set of unordered k-partitions of G, such that each block induces a connected subgraph. In this section, we will show that uniformly sampling from $\mathscr{P}_k(G)$ is intractable on the class of planar graphs. We will also show intractability for a family of probability distributions that weights partitions according to the size of their boundary.

2.6.1. Duality for connected k-partitions. Key to our proof will be a duality theorem Theorem 2.52, which is proven in detail in the appendix. We now list the definitions necessary for its statement.

DEFINITION 2.46 (Unordered connected partitions). *Let $\mathscr{P}^c(G)$ be the set of* unordered *partitions of $V(G)$ such that each block induces a connected subgraph. That is, $\mathscr{P}^c(G) = \bigcup_{k=1}^{|V(G)|} \mathscr{P}_k(G)$.*

DEFINITION 2.47 (Edge cut). *Let P be a partition of $V(G)$. If $P = \{A_1, \ldots, A_k\}$, then we refer to the A_i as the* blocks *of P. Let $\mathrm{cut}(P)$ denote the set of edges of G with endpoints in different blocks of P.*

DEFINITION 2.48 (Component map). *Given $J \subseteq E(G)$, define a partition $\mathrm{comp}(J) \in \mathscr{P}^c(G)$ as the partition into the connected components of $G \setminus J$.*

DEFINITION 2.49 (Dual connected partitions). *Let $\mathscr{E}_2(G)$ denote the set of subsets of edges of G such that each connected component of the induced subgraph is 2-edge connected:*

$$\mathscr{E}_2(G) = \{J \subseteq 2^{E(G)} : \text{ Each component of } G[J] \text{ is 2-edge connected}\}.$$

DEFINITION 2.50 (Circuit rank, number of components). *Let G be a graph. Then $h_1(G)$ denotes the circuit rank of G, that is, the minimum number of edges that must be removed to make G a forest. Also, $h_0(G)$ denotes the number of connected components of G.*

DEFINITION 2.51 (Dual k-partition). *We define $\mathscr{P}_k^*(G) = \{J \in \mathscr{E}_2(G) : h_1(G[J]) = k-1\}$. We call the elements of $\mathscr{P}_k^*(G)$ dual k-partitions.*

THEOREM 2.52 (Duality between $\mathscr{P}_k(G)$ and $\mathscr{P}_k^*(G^*)$). *Let G be a plane graph, and G^* its planar dual. Let $D : 2^{E(G)} \to 2^{E(G^*)}$ be the natural bijection. The map $D \circ \mathrm{cut} : \mathscr{P}_k(G) \to \mathscr{P}_k^*(G^*)$ is a bijection, with $\mathrm{comp} \circ D^{-1} : \mathscr{P}_k^*(G^*) \to \mathscr{P}_k(G)$ as its inverse. Both are computable in polynomial time.*

2.6.2. The corresponding NP-complete problem. We will show that it is NP-complete to decide if $\mathscr{P}_k^*(G)$ has length maximizing elements.

DEFINITION 2.53 (Spanning edge set). *Let J be a subset of edges of a graph G. We say that J* spans *G if every node of G is incident to some edge of J.*

PROPOSITION 2.54. *Let $G = (V, E)$ be a graph. The maximum number of edges any set $J \subseteq E$ with $h_1(G[J]) = k-1$ can have is $|V| + k - 2$. Moreover, a $J \subseteq E$ with $h_1(G[J]) = k-1$ has $|V| + k - 2$ edges if and only if $G[J]$ has one component and spans G.*

Proof. Let E_J, V_J be the number of edges and vertices of $G[J]$, respectively. From $E_J - V_J = h_1(J) - h_0(J)$ (Proposition A.21), we have $E_J - V_J = k - 1 - h_0(J)$. Thus, $E_J = V_J + k - 1 - h_0(J) \leq |V| + k - 2$, as $h_0(J) \geq 1$ and $V_J \leq |V|$. This establishes the upper bound. Moreover, these inequalities become equalities if and only if $V_J = |V|$ and $h_0(G[J]) = 1$, which is to say, if and only if J spans and has one component. ∎

To define the NP-complete decision problem we will use for the sampling intractability proof, we single out the elements of $\mathscr{P}_k(G)$ that achieve the upper bound of Proposition 2.54 and define a corresponding decision problem.

Definition 2.55 (Maximal dual k-partition). *Let $\mathscr{P}_k^*(G)_m$ be the subset of $\mathscr{P}_k^*(G)$ consisting of the subgraphs that have $|V(G)| + k - 2$ edges, which is the maximal number of edges possible.*

PlanarMaxEdgesDualkpartition
Input: A planar graph G
Output: YES if $\mathscr{P}_k^*(G)_m \neq \emptyset$, NO, otherwise.

We will prove that PlanarMaxEdgesDualkpartition is NP-complete, by reducing from the Hamiltonian cycle problem on grid graphs:

Theorem 2.56 ([59]). *Let G be a finite subgraph of the square grid graph $\mathbb{Z}^2$, wherein integer points are adjacent if and only if their Euclidean distance is 1. Deciding whether G has a Hamiltonian cycle is* NP*-complete.*

Proposition 2.57. *The problem* PlanarMaxEdgesDualkPartition *is* NP*-complete.*

Proof. The language of graphs that have $\mathscr{P}_k^*(G)_m \neq \emptyset$ is in NP, since checking if a given set of edges is in $\mathscr{P}_k^*(G)_m$ can be done in polynomial time. We now show the reduction from Hamiltonicity of grid graphs. Let G be a some subgraph of the grid graph, which we assume without loss of generality is 2-connected, since otherwise it has no Hamiltonian cycle. Because G is 2-connected, the lexicographic upper-left most node v of G must have degree 2. We build a new graph, G', by removing v and connecting its neighbors with a chain of $k-2$ diamonds, as in Figure 2.8.

We will show that G' has an element of $P_k^*(G)_m$ if and only if G has a Hamiltonian cycle. If G has a Hamiltonian cycle, that cycle had to pass through both edges of v, hence we can replace those edges with the diamonds in G'. The result has $h_1 = k-1$, spans G', and is connected. Thus it is an element of $P_k^*(G')_m$.

Going in the other direction, suppose that there is an $X \in \mathscr{P}_k^*(G')_m$. Since X must span G', X must contain each node in the chain of diamonds. Moreover, since elements of $\mathscr{P}_k^*(G)$ have no bridge edges (Lemma A.1), this implies that X contains all the edges in that chain of diamonds. Thus, $k-2$ of $h_1(X)$ is accounted for by the diamonds, and for $h_1(X) = k-1$, the rest of X must be a simple cycle, which spans $G \setminus v$ because X spans G'. Replacing those diamonds by the path $\{a, v, b\}$, we obtain a Hamiltonian cycle of G. ∎

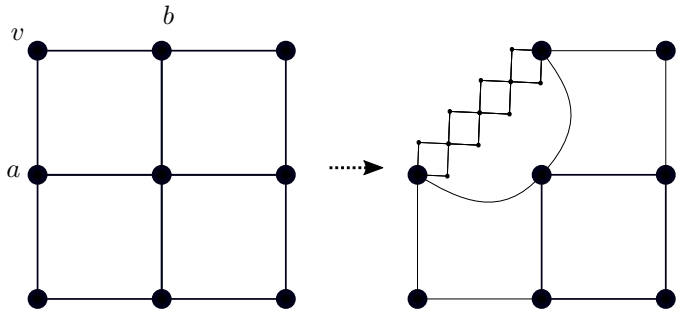
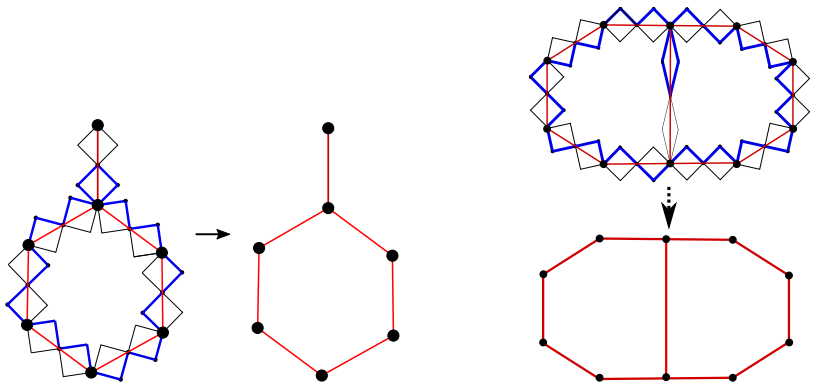


Figure 2.8: A step in the reduction in Proposition 2.57

2.6.3. The Chain of Dipoles Construction, Degeneration of k-Partitions. We will concern ourselves with the following sampling problem, where we fix $\lambda > 0$:

λ-sampling connected dual k-partitions
Input: A graph G.
Output: An element of $\mathscr{P}_k^*(G)$, drawn according to the probability distribution that assigns a set $J \in \mathscr{P}_k^*(G)$ weight proportional to $\lambda^{\|J\|}$.

To make some estimates, we give the probability distribution in this problem a name:



a) A degenerating dual 3-partition

b) A Dual 3-partition with a bigon from an $B_{r,d}$, but which does not degenerate

Figure 2.9: Example of degeneration and subtle non-degeneration.

DEFINITION 2.58 (Measures ν_λ and N_λ). *Let G be a graph, and let $\lambda > 0$. For any collection of sets of edges $Y \subseteq 2^{E(G)}$, we define the measure N_λ on Y by $N_\lambda(J) = \lambda^{|J|}$ for all $J \in Y$. We define a probability measure ν_λ by normalizing N_λ: $\nu_\lambda(J) = \frac{N_\lambda(J)}{N_\lambda(Y)}$.*

To prove that sampling from ν_λ is intractable, instead of using chains of bigons like we did for the uniform distribution, we will use chains of order-r dipoles, where r will be chosen so that $r\lambda \geq 2$:

DEFINITION 2.59 (Chain of order-r dipoles). *Define $B_{r,d}(G)$ as the graph obtained from G by subdividing each edge of G into d segments, and then replacing each edge of the resulting graph by r parallel edges (i.e. order-r dipoles). Let $B_{r,d}(e)$ denote the chain of d order-r dipoles that replaces the edge e.*

DEFINITION 2.60 (Dipole projection map). *We define a map $\pi_d : P_k^*(B_{r,d}(G)) \to 2^{E(G)}$ by $\pi_d(X) = \{e \in E(G) : X \cap B_{r,d}(e) \neq \emptyset\}$.*

Now we discuss the main technical hurdle to overcome in this section. Recall from §2 that when dealing with a simple cycle C, there was only one way that $\pi_d(C)$ could fail to be a simple cycle, namely, if C was one of the bigons. Crucially, there were only $d|E|$ ways this could happen, which was negligible compared to the size of the domain of π_d. On the other hand, elements of $P_k^*(G)$ can degenerate in more complicated ways, as is shown in Figure 2.9. The next few propositions establish that the degenerating elements—and all of the other possibilities—remain negligible compared to the preimages of $P_k^*(G)_m$:

LEMMA 2.61 (Surjectivity). *For each $Y \in P_k^*(G)$, there is a $\widetilde{Y} \in P_k^*(B_{r,d}(G))$ such that $\pi_d(\widetilde{Y}) = Y$.*

Proof. Simply replace each edge $e \in Y$ with a simple path of length d along $B_{r,d}(e)$. This does not change the topology, and hence the result has the same cycle rank and no bridges, which characterizes the elements of P_k^* by Lemma A.1. ∎

It will be convenient to single out the particular kind of π_d-preimage constructed in the proof of the last lemma:

DEFINITION 2.62 (Lift). *If $Y \in P_k^*(G)$, we refer to any of the $\widetilde{Y} \in P_k^*(B_{r,d}(G))$ obtained by replacing each edge $e \in Y$ with a simple path of length d through $B_{r,d}(e)$ as a* lift *of Y.*

To prove the probability concentration estimates (Lemma 2.67) we need to characterize the elements of $\mathrm{Im}(\pi_d)$ that have the most N_λ mass above them as the elements of $P_k^*(G)_m$. For that we will need to characterize the elements of the image with the most edges, which will be accomplished by Proposition 2.63 and Proposition 2.64.

PROPOSITION 2.63. *The elements in the image of $\pi_d : P_k^*(B_{r,d}(G)) \to 2^{E(G)}$ all have $h_1 \leq k-1$ and $\leq |V(G)| + k - 2$ edges.*

Proof. The bound on the number of edges will follow from Proposition 2.54 once we argue that all the elements in the image have $h_1 \leq k-1$. So, let $X \in P_k^*(B_{r,d}(G))$, and let $1_{C_1}, \ldots, 1_{C_m}$ be a basis for the cycle space of $\pi_d(X)$, where 1_Z is the indicator function of any set $Z \subseteq E(G)$. For each $i = 1, \ldots, m$, define $\widetilde{C}_i$ as a lift of C_i. We will show that the $1_{\widetilde{C}_i}$ are independent, by showing that any linear dependence gives a corresponding dependence between the 1_{C_i}. Set $E_d = E(B_{r,d}(G))$. Define a linear map $T : \mathbb{R}^{E_d} \to \mathbb{R}^{E(G)}$, which, for each $e \in E(G)$, adds up the values along all edges of $B_{r,d}(e)$ and sets that as the value of e. In particular, $T(1_{\widetilde{C}_i}) = d1_{C_i}$. Suppose that we had that $0 = \sum_{i=1}^m a_i 1_{\widetilde{C}_i}$, as functions on E_d. Then by applying T to this equation, we obtain $0 = \sum_{i=1}^m a_i d 1_{C_i}$, which implies that $a_i = 0$ for $i = 1, \ldots, m$. Hence $m \leq h_1(X)$, which implies that $m \leq k-1$, so $h_1(\pi_d(X)) = m \leq k-1$. ∎

PROPOSITION 2.64. *The elements of* $\mathrm{Im}(\pi_d)$ *that have* $|V(G)|+k-2$ *edges are the elements of* $P_k^*(G)_m$.

Proof. Suppose $K = \pi_d(X)$ has $|K| = |V(G)| + k - 2$. By Proposition 2.63, $h_1(K) \leq k-1$. By Proposition 2.54, $h_1(K) \geq k-1$, so $h_1(K) = k-1$. To finish the claim, we prove that K has no bridge edges. But suppose that e is a bridge edge, and let $\widetilde{e}$ be any edge in $B_{r,d}(e) \cap X$. X, having no bridges, must have a simple cycle C that contains $\widetilde{e}$. Now, let $C_1, \ldots, C_{k-1}$ be a cycle basis for K. Since e is a bridge edge, none of the C_i contain e, hence their lifts do not contain $\widetilde{e}$. This implies that C is not in the span of $C_1, \ldots, C_{k-1}$, hence $h_1(X) \geq k$, which contradicts $X \in P_k^*(B_{r,d}(G))$. This shows that every element in $\mathrm{Im}(\pi_d)$ with $|V(G)|+k-2$ edges is in $P_k^*(G)_m$. Since we already showed in Lemma 2.61 that every element of $P_k^*(G)$ has a lift to an element of $P_k^*(B_{r,d}(G))$, the claim follows. ∎

For the probability concentration lemma we will need the upper and lower bounds on the N_λ mass above an m edge element of $\mathrm{Im}(\pi_d)$ provided by the next two propositions:

PROPOSITION 2.65. *Let* $K \in P_k^*(G)_m$. *Then* $N_\lambda(\pi_d^{-1}(K)) \geq (r\lambda)^{d(|V(G)|+k-2)}$.

Proof. Since K has $|V(G)| + k - 2$ edges, and along each $B_{r,d}(e)$ there are r^d simple paths, there are $r^{d(|V(G)+k-2)}$ lifts obtained by choosing one of those paths for each edge. Since the length of each such lift is $d(|V(G) + k - 2|)$, the total N_λ mass is $r^{d(|V(G)|+k-2)}\lambda^{d(|V(G)+k-2)} = (r\lambda)^{d(|V(G)|+k-2)}$. Since there may be other preimages, as shown in Figure 2.9, we only get a lower bound. ∎

PROPOSITION 2.66. *Let* $K \in \mathrm{Im}(\pi_d)$, *with* $|E(K)| = m$. *Assume* $\lambda \in (0, 1]$ *and assume* r *is such that that* $r\lambda \geq 2$. *Then* $N_\lambda(\pi_d^{-1}(K)) \leq 2^m r^{2km} d^{2km} (r\lambda)^{dm}$.

Proof. We begin by bounding the number of possible configurations of $X \cap B_{r,d}(e)$ for any $e \in E(G)$, and $X \in P_k^*(B_{r,d}(G))$. First, observe that there are r^d simple paths across $B_{r,d}(e)$. We treat two cases, depending on whether or not X contains one of these paths.

If X contains one of those paths then X may contain at most $k-1$ additional edges from $B_{r,d}(e)$, because each one increases the rank of h_1. Thus, $r^d\binom{rd}{k-1}$ upper bounds the number of configurations which includes a path through $B_{r,d}(e)$. Moreover, each such configuration contains at least d edges, hence each one has N_λ mass at most λ^d, as $\lambda \leq 1$.

Alternatively, $X \cap B_{r,d}(e)$ may not contain any of the simple paths crossing $B_{r,d}(e)$. However, in this case, we have that $|X \cap B_{r,d}(e)| \leq 2(k-1)$, since every pair of edges will increase h_1 by one, and thus an upper bound on the number of such configurations is $\binom{rd}{2(k-1)}$. Moreover, each one of these configurations contains at least 2 edges, so has mass at most λ^2. Thus, the total N_λ mass obtained from this case is $\binom{rd}{2(k-1)}\lambda^2$.

Combining these two cases, we have a bound for the total N_λ mass of $\pi_d^{-1}(K)$ restricted to $B_{r,d}(e)$, namely:

$$\sum_{X \in \pi_d^{-1}(K)} N_\lambda(X \cap B_{r,d}(e)) \leq \binom{rd}{k-1} r^d \lambda^d + \binom{rd}{2(k-1)} \lambda^2 \leq 2(rd)^{2k}(r\lambda)^d.$$

Here, the last inequality follows from $\max(\binom{rd}{k-1}, \binom{rd}{2(k-1)}) \leq (rd)^{2k}$ and from $r\lambda \geq 2 \geq \lambda^2$. To determine $N_\lambda(\pi_d^{-1}(K))$, we first observe that $N_\lambda(X) = \lambda^{|X|} = \prod_{e \in E(K)} \lambda^{|X \cap E(B_{r,d}(e))|} = \prod_{e \in E(K)} N_\lambda(X \cap B_{r,d}(e))$ for

any $X \in \pi_d^{-1}(K)$. From this, the result follows:

$$\begin{aligned} N_\lambda(\pi_d^{-1}(K)) &= \sum_{X \in \pi_d^{-1}(K)} N_\lambda(X) \\ &= \sum_{X \in \pi_d^{-1}(K)} \prod_{e \in E(K)} N_\lambda(X \cap B_{r,d}(e)) \\ &\leq \prod_{e \in E(K)} \sum_{X \in \pi_d^{-1}(K)} N_\lambda(X \cap B_{r,d}(e)) \\ &\leq \prod_{e \in E(K)} 2(rd)^{2k}(r\lambda)^d \\ &= 2^m (rd)^{2km} (r\lambda)^{dm} \end{aligned}$$

∎

We now prove the probability concentration lemma necessary for applying the lucky guess algorithm, Lemma 2.8:

LEMMA 2.67 (Probability concentration lemma). *Let $G = (V, E)$ have $n = |V|$. Let $\lambda \in (0, 1]$ and fix an integer r such that $r\lambda \geq 2$. Assuming that $P_k^*(G)_m$ is non-empty, then for $d = 4\lceil(\frac{a+2n^2+2kn^2(\log(r)+1)}{\log(r\lambda)})^2\rceil$ the probability under ν_λ that an element of $P_k^*(B_{r,k}(G))$ maps via π_d to an element of $P_k^*(G)_m$ is at least $\frac{2^a}{1+2^a}$.*

Proof. Let $M = n + K - 2$. Since elements of $\mathrm{Im}(\pi_d) \setminus P_k^*(G)_m$ have $\leq M - 1$ edges (Proposition 2.64), and $|Im(\pi_d)| \leq 2^{n^2}$, we have $N_d := N_\lambda(\pi_d^{-1}(2^{E(G)} \setminus P_k^*(G)_m)) \leq 2^{n^2} 2^{M-1} (rd)^{2k(M-1)} (r\lambda)^{d(M-1)}$ (Proposition 2.66). We also have that $H_d := N_\lambda(\pi_d^{-1}(P_k^*(G)_m)) \geq (\lambda r)^{dM}$ (Proposition 2.65). Using Lemma 2.10 and $M \leq n^2$, for $d \geq 4(\frac{\log(S)+2kn^2}{\log(q)})^2$, where $S = 2^a 2^{2n^2} r^{2kn^2}$ and $q = r\lambda$, we have that $H_d \geq 2^a N_d$. Hence, by Lemma 2.9, this d suffices for $\frac{H_d}{N_d + H_d} \geq \frac{2^a}{1+2^a}$. ∎

2.6.4. Uniformly sampling connected k-partitions is intractable. In this section we prove intractability of sampling dual k-partitions, and connect this to the intractability of sampling connected k-partitions.

THEOREM 2.68. *For any fixed $\lambda \in (0, 1]$, λ-sampling connected dual k-partitions is intractable on the class of 2-connected planar graphs.*

Proof. We will assemble the ingredients for Lemma 2.8, the Lucky Guess lemma. We fix a choice of a so that $\frac{2^a}{2^a+1} \geq 1 - 1/m$. Define B is by the construction $G \to B_{r,d}(G)$ where r is chosen so that $r\lambda \geq 2$ and $d = \lceil(\frac{a+2n^2+2kn^2(\log(r)+1)}{\log(r\lambda)})^2\rceil$. The map M is given by π_d, where we have $S(G) = 2^{E(G)}$. The problem Q is defined by $Q(G) = P_k^*(G)_m$, which is NP-complete by Proposition 2.57. Moreover, by Lemma 2.67, we have that $\mathbb{P}(\pi_d(C) \in Q(G) : C \text{ is distributed according to } \nu_\lambda \text{ on } P_k^*(B_{r,d}(G)) \geq 1 - 1/m$. Thus, we obtain the result from Lemma 2.8. ∎

For any fixed $\lambda > 0$, we define a distribution on connected k-partitions.

λ-SAMPLING CONNECTED k-PARTITIONS
Input: A graph G.
Output: An element of $\mathscr{P}_k(G)$, drawn according to the probability distribution that assigns a partition $P \in \mathscr{P}_k(G)$ weight proportional to $\lambda^{\lvert\mathrm{cut}(P)\rvert}$.

Now we state the main theorem of this section:

THEOREM 2.69. *Fix $\lambda \in (0, 1]$. Then λ-sampling connected k-partitions is intractable on the class of 2-connected planar graphs.*

Proof. This follows as a corollary to Theorem 2.68, using Theorem 2.52. ∎

3. The Flip Markov Chain. In the previous section, we examined the worst case complexity of the partition sampling problem. However, worst case intractability results do not necessarily mean that the problem is intractable on examples of interest, since there can be algorithms which are effective only on certain cases. In this section and the next, we examine the performance of one such algorithm, which is based on Markov chains.

Markov chains provide a generic means of sampling from prescribed distributions over a state space Ω. This technique starts with a seed in Ω and randomly applies perturbations to walk around the space; the more steps in the random walk, the closer the sample is to being distributed according to the *stationary* distribution of the chain rather than the seed point. While this approach provides an elegant means of sampling, a mathematical analysis of the *mixing time* (see §3.3) is needed to understand how many steps one must take before the output can be trusted as representative of the stationary distribution. Without control over the mixing time, it is possible that the sample did not travel far from the initial seed, potentially yielding a biased sample and distorted measurements.

In this section, we discuss a commonly-used Markov chain for sampling from $P_2(G)$, which we call the *flip walk*. This chain has seen wide use in the analysis of gerrymandering [16, 31, 77]. We know from our analysis in the previous sections that one cannot hope for this chain to mix rapidly on general graphs, unless one also believes that RP = NP. To make this more concrete in this section we show that the gadgets used in our complexity proofs directly yield bottlenecks impeding the mixing of the flip chain. Later, in §4, we will use ideas from this section to analyze the mixing of the flip chain on examples relevant to redistricting.

3.1. Related work. The flip walk is analogous to Glauber dynamics and Potts models. Contiguity of the blocks is not usually considered in these physical settings, and it is part of what makes sampling districting plans challenging. A difficulty in analyzing the combinatorics of contiguity constraints is that it is defined through global, rather than local interactions; a physical model with similar challenges is the self-avoiding walk, which we consider further in (§4.1). We now list a few works that have studied Markov chains similar to the flip walk:

3.1.1. Sampling min-cuts, and cuts according to boundary length. A Markov chain with similar proposal moves as in Definition 3.1 was studied in [21], but restricted to a state space of st-cuts instead of connected k-partitions. They show that this Markov chain mixes slowly, even if the underlying graph is of bounded treewidth. They prove a tree-width fixed parameter tractability results for the counting and sampling problems they consider which, similarly to our work in §5, build on Courcelle's theorem and are based on dynamic programming. Their results therefore share some similarities with ours, except that we studied *connected* 2-partitions weighted by a function of the edge-cut, whereas they studied two different cases: min cuts, and *all* cuts weighted by a function of the edge-cut. The example in their section 7.4 has some similar features to our example in Figure 3.2.

3.1.2. Literature on sampling st-paths. Another place in the literature where the flip walk appears is in [78], where the problem of sampling simple st-paths using the flip walk is studied. Their paper provides another example where there is a bottleneck [78, Theorem 7] and gives a proof of ergodicity for their version of the flip chain (which is restricted to st-paths for fixed s and t). They also make the observation that if the st-path flip chain on the grid graph is restricted to paths that are *monotone* in one direction, then the flip walk is rapidly mixing on that restricted state space. We remark that the techniques based on [63, Prosition 5.1] that we discussed in §2 suffice to show that the sampling problem they consider is intractable on any class of graphs closed under the operation of replacing edges with chains of bigons, and where the Hamiltonian st-paths problem is NP-complete. Additionally, the techniques we discuss below in §5 should suffice to reduce the simple st-path sampling problem to a corresponding counting problem, which will be tractable on certain classes of graphs, such as series parallel graphs or graphs of bounded treewidth.

The question of sampling simple paths has also received some attention in the literature: [56] proves that a certain Markov chain on simple paths in a complete graph mixes rapidly (Theorem 4.1.2) but that a Metropolis-Hasting's version with weights has bottlenecks (Theorem 4.2.2), and repeats a similar analysis for sampling simple paths in trees (Theorem 4.3.2 and Theorem 4.4.1). The existence of a FPRAS for weighted simple paths on the complete graph, where weights can be set to zero to forbid edges, would imply the existence of a FPRUS for simple paths in any graph, which would imply that RP = NP by using the

chain of bigons trick from [63, Prosition 5.1] and the NP-completeness of the Hamiltonian path problem; this negatively answers one of the open problems given in the conclusion of [56]. [56] also provides a dynamic program algorithm to count and uniformly sample weighted simple paths in trees and DAGs (Section 5). The undirected case can be extended using Courcelle's theorem, so it is likely that a reasonably implementable fixed-parameter in treewidth algorithm for sampling simple paths exists, perhaps along similar lines to §5.

3.2. The Flip Walk. We put a graph structure on the set of connected 2-partitions $P_2(G)$ as follows. Let $(A, B) \in P_2(G)$. Given any $x \in V(G)$, consider the partition $(A \cup \{x\}, B \setminus \{x\}) = (A', B')$. Provided that $(A', B') \in P_2(G)$, including the case when $(A', B') = (A, B)$, this defines an *edge* between two elements of $P_2(G)$. If $(A', B') \notin P_2(G)$, that is, if either A' or B' does not induce a connected subgraph of G, then we add a self loop to (A, B). Do the same also for $(A \setminus \{x\}, B \cup \{x\})$. This defines a $|V(G)|$-regular graph structure on $P_2(G)$. Given this graph structure, we define the flip walk:

Definition 3.1 (Flip Walk). *The flip walk on $P_2(G)$ is the Markov chain obtained by performing a lazy simple random walk on $P_2(G)$, using the graph structure defined in the previous paragraph. We abuse notation and refer to the Markov chain, the graph, and the set by $P_2(G)$.*

If G is 2-connected, then $P_2(G)$ is irreducible [12] and hence ergodic. Since every node of $P_2(G)$ has degree $|V(G)|$, the uniform distribution is the stationary distribution for the flip walk on $P_2(G)$. Thus, by standard Markov chain theory [70], this flip walk *eventually* produces nearly uniformly distributed elements in $P_2(G)$. However, we will see examples in this section where the flip walk on $P_2(G)$ can take exponential time in $|G|$ to generate a nearly uniform sample.

3.3. Background: Mixing time of Markov Chains. We make a short digression to review a few notions from Markov chain theory. For details we have left out, see [70]. Since the goal of our discussion is to give examples where the random walk on $P_2(G)$ mixes slowly, we will recall the notion of *mixing time* in the context of (discrete) Markov chains:

Definition 3.2 (Total variation). *Given two probability distributions μ and ν on a finite set Ω, the* total variation *distance between μ and ν is given by $\|\mu - \nu\|_{TV} = \frac{1}{2}\sum_{x\in\Omega} |\mu(x) - \nu(x)|$.*

Definition 3.3 (Mixing time). *Let μ be the stationary distribution of the (discrete time) Markov chain $M = (\Omega, P)$. Let $P^t\delta_x$ denote the distribution at time t of the Markov chain M started at x. Define*

$$d^M(t) := \max_{x\in\Omega} ||P^t\delta_x - \mu||_{TV}.$$

Then, the mixing time *of M is*

$$t^M_{mix}(\epsilon) = \inf\{t : d^M(t) \leq \epsilon\}. \tag{3.1}$$

If the chain is clear from the discussion, we omit the superscript M.

The definitions above help formalize what it means for a Markov chain to mix rapidly or torpidly:

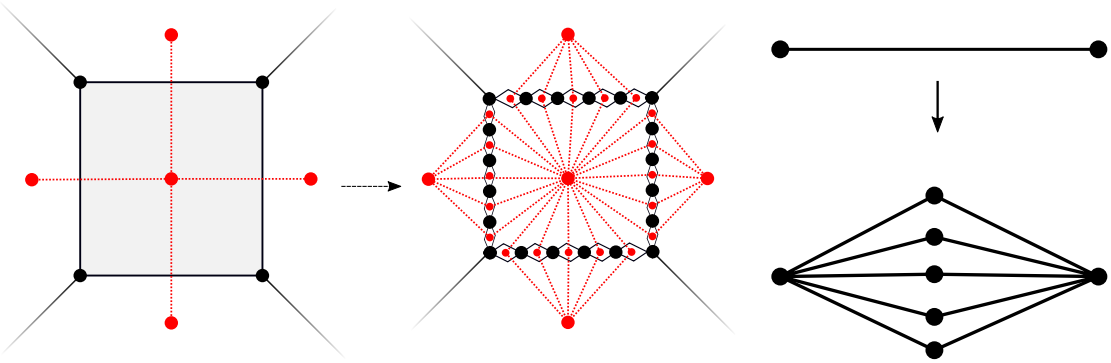
Definition 3.4 (Rapidly mixing). *A family of Markov chains $M \in \mathcal{M}$ is said to be* rapidly mixing *if the there is a polynomial $p(x, y)$ so that $t^M_{mix}(\epsilon) \leq p(\log|M|, \log\epsilon)$, $\forall M \in \mathcal{M}$, where $|M|$ denotes the size of the state space of M.*

To prove rapid mixing, it is equivalent to find a polynomial $q(x)$ so that $t^M_{mix}(1/4) \leq q(\log(|M|))$, $\forall M \in \mathcal{M}$, as $t^M_{mix}(\epsilon) \leq \lceil\log_2(\epsilon^{-1})\rceil t^M_{mix}(1/4)$ [70, Equation (4.36)].

Definition 3.5 (Torpidly mixing). *If there is an exponentially growing function, $f(n)$, such that $t^M_{mix}(1/4) \geq f(\log(|M|))$, for all $M \in \mathcal{M}$, then we say that M is torpidly mixing.*

A standard means of arguing about mixing times for random walks on regular graphs comes from measuring bottlenecks, as in the next definition:

Definition 3.6 (Conductance). *Let G be a d-*regular *graph, and M the Markov chain obtained by a lazy*



(a) Illustrating that $D_d(H) = (B_d(H^*))^*$ for plane H (b) Doubled d-star

Figure 3.1: The chain of bigons construction and its dual

random walk on G. We define the conductance of M[8] *as*

$$\Phi(M) = \min_{\substack{U \subset V(G) \\ |U| \leq \frac{1}{2}|V(G)|}} \frac{|\partial_E U|}{2d|U|}.$$

Loosely speaking, such a set U which proves that $\Phi(M)$ is small is called a bottleneck.

The following theorem connects mixing time and conductance and will be used to show that the chain $P_2(G)$ mixes torpidly for certain families of graphs, by building explicit bottleneck sets that upperbound the conductance:

THEOREM 3.7 ([70]). *For every Markov chain M, $t^M_{mix}(1/4) \geq \frac{1}{4\Phi(M)}$.*

3.4. Bottlenecks from the chain of bigons construction. Due to the sampling intractability results (§2), we know that by replacing edges with chains of bigons, we created graphs whose simple cycles should be expensive to sample. Likewise, it should be expensive to sample the connected 2-partitions of the plane duals of these graphs. It is therefore natural to look for bottlenecks in the flip walk that arise through the plane dual of the chain of bigons construction (Figure 3.1(a)). In this section, we describe the dual of the chain of bigons construction (Definition 3.8) and explain how it creates bottlenecks.

DEFINITION 3.8 (Doubled d-star, original nodes). *Let H be a graph. The* doubled d-star *construction applied to H, notated $D_d(H)$, is obtained by replacing each edge of H with d parallel edges and then subdividing each new edge once. For $e \in E(G)$, we will let $D_d(e)$ denote the doubled d-star subgraph that replaced it. See Figure 3.1(b) for an illustration.*

There is an obvious inclusion $V(G) \hookrightarrow V(D_d(G))$, and we call the nodes in the image of that inclusion the original vertices. The other vertices in $D_d(G)$ are called new vertices.

We will find bottlenecks in $P_2(D_d(G))$ by relating it to $P_2(G)$ using Lemma 3.1:

LEMMA 3.1. *For any $(A, B) \in P_2(D_d(G))$, $(A \cap V(G), B \cap V(G)) \in P_2(G)$.*

Proof. Let $x, y \in V(G)$ be members of the same block of (A, B), say A. There exists a path γ in A from x to y. This path alternates between new vertices and original vertices. Forgetting the new vertices in this path gives a path in $A \cap V(G)$ between x and y. ∎

We use Lemma 3.1 to make the following definition:

DEFINITION 3.9 (Restriction map). *Define a map $R_d : P_2(D_d(G)) \to P_2(G)$ by setting $R_d((A, B)) = (A \cap V(G), B \cap V(G))$.*

[8] This is not the usual definition of the conductance, but this is the correct formula for the conductance of a lazy random walk on a d-regular graph [70, p. 144]. The formula there has a typo, which was corrected in the errata.

We now explain the key intuition behind the bottlenecks. In order for the flip walk to move between the fibers of R_d—that is, to change the assignment of an old node—a certain rare event must occur. In particular, if u and v are adjacent old nodes, and $u \in A$ and $v \in B$, then to reassign u to B, every new node in $D_d(\{u, v\})$ must already be in B. However, under the flip walk with $u \in A$, $v \in B$, the new nodes of $D_d(\{u, v\})$ behave like a random walk on a hypercube, and in particular, it is unlikely for them to become part of the same block. Pursuing this intuition, the next lemma proves that the fibers of R_d have *much* smaller edge boundary than size, which will mean that they are bottleneck sets:

LEMMA 3.10. *Suppose that $(A, B) \in P_2(G)$, with $A \neq \emptyset$ and $B \neq \emptyset$, and let $n = |V(G)|$. Then,*

$$|R_d^{-1}((A, B))| = 2^{\mathrm{cut}(A,B)d} \tag{3.2}$$

and

$$|\partial_E R_d^{-1}((A, B))| \leq (d + 1)n2^{(\mathrm{cut}(A,B)-1)d}. \tag{3.3}$$

Proof of (3.2). We will count the number of extensions of (A, B) across the new nodes of $D_d(G)$, by considering each edge $e \in E(G)$ separately. If $e \in \mathrm{cut}(A, B)$, then one can assign new nodes of $D_d(e)$ arbitrarily without affecting contiguity, and therefore one has 2^d choices. If $e \notin \mathrm{cut}(A, B)$, suppose both endpoints of e are in A. Since $B \neq \emptyset$, to preserve connectivity all the new nodes of $D_d(e)$ must be in A. Therefore, there is only one choice for how to extend (A, B) along this edge. Combining these two cases yields (3.2). ∎

Proof of (3.3). Let $e \in \partial_E R_d^{-1}((A, B))$ be an edge between $(L, M), (L', M') \in P_2(D_d(G))$, with $R_d((L', M')) =: (A', B')$. There is an $x \in V(G)$ be such that $L' = L + x$, $M' = M - x$, $A' = A + x$ and $B' = B - x$. Since $L \neq \emptyset$, for L' to be connected there has to be at least one node $l \in L$ so that $l \sim x$. Moreover, since $M - x$ is connected and $l, x \notin M - x$, $M - x$ can contain at most one new node of $D_d(\{l, x\})$. Hence, there are at most $d+1$ extensions of (A', B') onto the new nodes of $D_d(\{l, x\})$. As there are most $\mathrm{cut}(A, B) - 1$ edges remaining where we might have the full 2^d range of extensions, it follows that there are *at most* $(d + 1)2^{(\mathrm{cut}(A,B)-1)d}$ elements of $\partial_E R_d^{-1}((A, B))$ that map to $\{(A, B), (A', B')\}$ under R_d. Finally, the claim follows because there are at most n candidates for the original node x that gets flipped when making a step across $\partial_E R_d^{-1}(A, B)$. ∎

We now use these computations to show the slow mixing of the flip chain.

THEOREM 3.11. *Let G be any 2-connected graph with at least two distinct connected 2-partitions $P, Q \in P_2(G)$, neither of which have the empty set as a block. Let $n = |V(G)|$. Then, the family $P_2(D_d(G))$, $d \geq 1$, is torpidly mixing. In particular, we have the following bounds:*

$$\Phi(P_2(D_d(G))) \leq (d + 1)2^{-d-1}, \tag{3.4}$$

$$t_{mix}(1/4)(D_d(G)) \geq \frac{2^{d-1}}{(d + 1)}, \text{ and} \tag{3.5}$$

$$|P_2(D_d(G))| \leq 2^{|D_d(G)|} \leq 2^{n+dn^2}. \tag{3.6}$$

Proof. Without loss of generality, assume that $\mathrm{cut}(P) \leq \mathrm{cut}(Q)$. We have that $|R_d^{-1}(P)| \leq 1/2|P_2(D_d(G))|$ since $R_d^{-1}(P) \cap R_d^{-1}(Q) = \emptyset$ and $|R_d^{-1}(Q)| \geq |R_d^{-1}(P)|$ by (3.2). Combining (3.2) with (3.3) yields (3.4). Equation (3.5) follows from this by Theorem 3.7. Finally, (3.6) follows from the construction of $D_d(G)$. From (3.6), we have that $\log |P_2(D_d(G))| \leq n + dn^2$.

For a concrete example, take G to be a 4-cycle, and take the two 0-balanced cuts. A snapshot of the evolution of the flip walk on $D_5(G)$ can be seen in Figure 3.2.

We pause to note a key division between the intractability of uniformly sampling $P_2(G)$ and the mixing of the flip walk. First, observe that if G is a series-parallel graph , then $D_d(G)$ is series parallel as well. We show in §5 that there is a polynomial time algorithm to uniformly sample from $P_2(G)$ on the class of series-parallel graphs, and yet our proof above shows that the flip walk still mixes slowly on this class of graphs. Thus, even in cases where uniform sampling is tractable, the flip walk on $P_2(G)$ still may not be an efficient means of sampling.

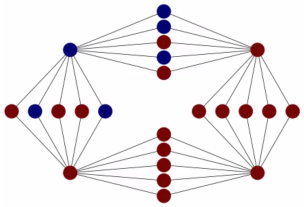


Figure 3.2: A snapshot of the flip walk evolving, illustrating the bottleneck of Theorem 3.11.

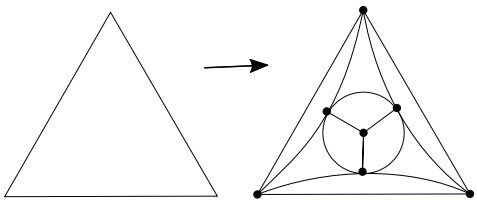


Figure 3.3: The affect of T_2

3.5. Bottlenecks from the R_d construction. The example of Theorem 3.11 is not entirely satisfying, for example because the degrees of its nodes increase without bound. We will address some of its weaknesses in this section by producing a family of maximal plane graphs with vertex degree ≤ 9, such that the corresponding family of flip walk chains is torpidly mixing. As in §3.4, our strategy will be to find a construction $G \to T_d(G)$ which uses gadgets to refine certain features of G, and a map $P_2(T_d(G)) \to P_2(G)$, where we can count the size of the fibers and the size of the edge boundaries of the fibers. All graphs in this section are assumed to be embedded in the plane. Additionally, we will freely describe a partition $(A, B) \in P_2(G)$ by a map $p : V(G) \to \{a, b\}$.

Definition 3.12 (T_d, original vertices, original triangles). *Let G be a maximal plane graph. Let $T_d(G)$ be the graph defined by $T_d(G) = (R_d(G^*))^*$, where R_d is as in Definition 2.34. There is a natural injection $i : V(G) \to V(T_d(G))$, since there is a natural injection $Faces(G^*)$ to $Faces(R_d(G^*))$, and we call the nodes in $\mathrm{Im}(i)$ the original vertices. Moreover, if F is any triangular face in G, then we call the original vertices of F in $V(T_d(G))$ an original triangle.*

The affect of T_d is to take every triangular face and refine it by gluing a graph hanging from the three nodes of the triangle. Figure 3.3 shows the affect of applying T_2 on a single triangular face.

In §3.4, our strategy for finding bottleneck sets was to find a set of vertices such that if they all belonged to the same block, then a large number of vertices would also belong to that block. In this section, such a set of vertices will be the vertices of an original triangle. If all the nodes of an original triangle are in the same block, we will call that triangle pure:

Definition 3.13 (Pure and mixed faces). *Let G be a plane graph. Consider a partition $(A, B) \in P_2(G)$ defined by a function $p : V(G) \to \{a, b\}$. We will call a face F* pure *of assignment a (resp. b), if p takes the value a (resp. b) on all of its nodes. We will call the face* mixed *otherwise, that is, if p takes on both values on the vertices of F. For a partition $(A, B) \in P_2(G)$, we let $\mathfrak{P}_{(A,B)}$ be the function on the set of faces of G that assigns a to all pure a-faces, b to all pure b-faces, and m to all mixed faces. Additionally, we define $M : P_2(G) \to \mathbb{N}$ as the number of mixed faces in a partition.*

We are going to find bottlenecks in this section by defining sets of partitions of $T_d(G)$ by whether all original triangles are mixed. To leave such a set, some of the triangles will have to become pure, which will force the new nodes of that triangle to be in a specific arrangement. A convenient tool for expressing this will be to describe directed edges of $P_2(G)$ as being purifying or not.

Definition 3.14 (Directed Configuration Space). *For a graph G, let $DP_2(G)$ be the directed graph version of the flip walk adjacency structure on $P_2(G)$. That is, it has a node for each node of $P_2(G)$ and for*

each edge $e = \{P, Q\}$ in $P_2(G)$, $DP_2(G)$ has two edges: (P, Q) and (Q, P).

DEFINITION 3.15 (Purifying edges of the directed configuration space). *We call an edge $e = (P, Q) \in DP_2(G)$ purifying if there is a face F of G so that $\mathfrak{P}_P(F) = m$ but $\mathfrak{P}_Q(F) \in \{a, b\}$. Let $DP_2^C(G)$ be the graph obtained from $DP_2(G)$ by removing all purifying edges. For $Q \in P_2(G)$, we let $C_Q \subseteq P_2(G)$ be the set of all vertices strongly reachable from Q in $DP_2^C(G)$.*

Finally, we will need a way to relate connected 2-partitions of $T_d(G)$ to those of G, so that we can partition $P_2(T_d(G))$ based on which faces of G are pure or mixed.

LEMMA 3.2. *For any partition $(A, B) \in P_2(T_d(G))$ defined by $p : V(T_d(G)) \to \{a, b\}$, let $p_o : V(G) \to \{a, b\}$ denote the restriction to the original vertices, which we identify with $V(G)$. Then $p_o : V(G) \to \{a, b\}$ defines a connected partition of G.*

Proof. Let $x, y \in i(V(G)) \cap A$. Since $T_d(G)[A]$ is connected, there exist a path γ in A from x to y. Forgetting the new vertices in this path gives a path in $G[i^{-1}(A)]$, since all the vertices on the boundary of any original triangle are adjacent. ∎

DEFINITION 3.16 (Restriction map). *Define $F_d : P_2(T_d(G)) \to P_2(G)$ to be the restriction map $F_d(p) = p_o$, with notation as in Lemma 3.2*

LEMMA 3.17. *Let G and F_d be as above. Let $P \in P_2(G)$. Then,*

$$|F_d^{-1}(P)| \geq 5^{dM(P)} \tag{3.7}$$

and, supposing additionally that P is such that $C_P = \{P\}$,

$$|\partial_E F_d^{-1}(P)| \leq n5^{(d+1)(M(P)-1)}. \tag{3.8}$$

Proof of (3.7). A mixed face of a 2-partition of $T_d(G)$ corresponds to a $SBL(R_d)$ segment of the simple cycle dual to that 2-partition. Using the estimates in Equation (2.7), each of the mixed faces can be in at least in 5^d configurations, and so the claim follows. ∎

Proof of (3.8). Since $C_P = \{P\}$ any edge out of $F_d^{-1}(P)$ must cause one mixed face of P to become pure. This mixed face must be in a configuration where all but one node has the same block assignment, and that one exceptional node must be an original node. Since there are at most n original nodes of G which can switch during this step, and the other mixed faces have at most 5^{d+1} configurations each, the result follows by the bound on SBL from Equation (2.7). ∎

Taking $G = K_4$ yields the following corollary:

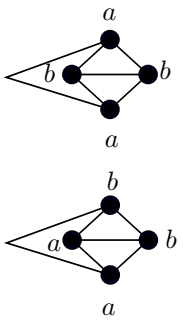
COROLLARY 3.18. *There is a family of graphs H_d, $d \in \mathbb{N}$, that are triangulations of the plane (maximal planar graphs) such the vertex degree is bounded by 9 and $|V(H_d)| = O(d)$, and for which $P_2(H_d)$ and the unordered partition chain have mixing times at least $\frac{5^d}{250}$. H_2 is shown in Figure 3.4b).*

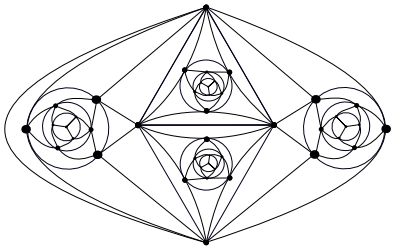
Proof. One can compute $DP_2^C(K_4)$ to find that there are three $P_i \in P_2(K_4)$ with $C_{P_i} = \{P_i\}$. Figure 3.4a) shows two such examples. Let P be the top partition and Q the bottom one in Figure 3.4a). By symmetry, we have that $|F_d^{-1}(P)| = |F_d^{-1}(Q)|$, and so since $F_d^{-1}(P) \cap F_d^{-1}(Q) = \emptyset$, it follows that $|F_d^{-1}(P)| \leq {}^{|P_2(T_d(G))|}/_2$. Hence, $F_d^{-1}(P)$, is a candidate bottleneck set. We compute,

$$\frac{|\partial_E F_d^{-1}(P)|}{2n|F_d^{-1}(P)|} \leq \frac{1}{2}5^{(d+1)(M(P)-1)-dM(P)}.$$

As $M(P) = 4$, it follows that $\Phi(P_2(T_d(G)) \leq \frac{1}{2}(5^{3-d})$. We obtain corresponding bottleneck sets in the quotient chain of unordered partitions. The result now follows by Theorem 3.7. ∎

This last example illustrates that controlling neither the degree, nor the face degree, nor insisting on 3-connectedness of G can improve the mixing time of the flip walk on $P_2(G)$. On the other hand, these graphs still have a lot of area enclosed by length 3 loops, which is arguably unrealistic for redistricting, except we could in principle see similar behavior around very dense cities. In the next section, we will use statistics inspired by the idea that certain nodes may change their assignment infrequently, as well as literature on self avoiding walks, to investigate the flip walk on connected partitions of a grid graph and on state dual graphs.





a) The elements of $DP_2^C(K_4)$ used in Corollary 3.18 b) H_3 of the family of Corollary 3.18.

Figure 3.4

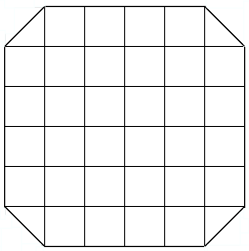


Figure 4.1: L_n

4. Empirical Examples. The torpid mixing of the flip walk highlighted in the previous section is not only a theoretical observation. In this section, we present several experiments showing slow mixing in practical applications of the flip walk to redistricting. The key statistic we study is closely linked to our bottleneck proofs, wherein we were able to identify sets of nodes that flip infrequently. For an empirical analysis, we will observe the frequency at which nodes flip during simulations of the flip walk on lattice like graphs (§4.2) and on state dual graphs (§4.2.1).

To put our investigation in a larger context, we will begin by reviewing some related to self avoiding walks on the grid graph (§4.1). This review will lead us to investigate the impact of the graph topology on the output the flip walk (§4.3), and on the output of another popular partition sampling algorithm (§4.3.1).

These experiments used the open source graph partition Markov chain software Gerrychain. The code that produced these experiments is available online [1].

4.1. Grid Graph. In this section, we will review some special features of connected partitions of the grid graph, in particular through the connection to the self-avoiding walk model from statistical physics.

4.1.1. Self-avoiding walks. In one special case, the objects we are studying are closely linked to a famous topic from statistical physics, namely self-avoiding walks on lattices. A self-avoiding walk (resp. polygon) is a simple path (resp. cycle) in the integer lattice graph. These were introduced as models for polymers, and have shown themselves to be a difficult and rich object of mathematical investigation. An excellent reference for this topic is [74]; [87] gives an overview of Monte Carlo methods used to investigate this topic. Self-avoiding walks on other lattices are also of interest, and we discuss one of those below. We will primarily be interested in walks that are constrained to lie in certain subsets of the lattice.

Take L_n to be the grid graph with shaved corners, as in Figure 4.1. The dual graph, L_n^*, is an $n-1 \times n-1$ grid graph G_{n-1} with an additional "supernode" V corresponding to the unbounded face. The simple cycles of L_n^* break into two classes. First, there are those that do not contain the supernode. These can be thought

of as self-avoiding polygons in G_{n-1}. Second, there are those that do contain the supernode. We can think of these as self-avoiding walks in G_{n-1} between two points on the boundary. We will call these *chordal* self-avoiding walks.

This connection to self-avoiding walks is important to us because such self-avoiding walks display phase transitions as one varies the preference for longer or shorter walks. As we will see, these phase transitions persist into the distribution ν_λ on 2-partitions that we studied in Definition 2.58. After recalling the relevant facts and history about self-avoiding walks, we will present our experiments.

DEFINITION 4.1 (The family P_λ of probability distributions on self-avoiding walks). *Fix $\lambda > 0$. Given a finite set of self-avoiding walks, P_λ is a probability distribution that assigns mass to each walk ω proportionally to $x^{|\omega|}$. Here $|\omega|$ counts the number of edges in the walk.*

It is known that the geometry of P_λ-typical chordal self-avoiding walks in G_n display a phase transition as the parameter λ is varied: depending on λ, for sufficiently large n, a path drawn from the distribution P_λ will have certain properties with high probability.[9] To state this phase transition, we will recall an important constant called the connective constant of the square lattice.

THEOREM 4.2 ([74], Connective constant of the square lattice). *Let c_n be the number of self-avoiding walks on the lattice $\mathbb{Z}^2$ that start at the origin. The limit $\mu = \lim_{n\to\infty} \sqrt[n]{c_n}$ exists. μ is called the connective constant of the square lattice, and $\mu \approx 2.683$.*

The phase transition for the qualitative properties of P_λ on chordal self-avoiding walks (SAW) occurs at $\lambda = 1/\mu$, which is called the critical "fugacity". In particular:

- Subcritical fugacity $\lambda < 1/\mu$: A P_λ-typical SAW resembles a geodesic on the grid graph [43].
- Critical fugacity $\lambda = 1/\mu$: A P_λ-typical SAW resembles a sample path from chordal $SLE_{8/3}$ [43, 69].
- Supercritical fugacity $\lambda > 1/\mu$: A P_λ-typical SAW is "space filling", in a sense made precise in [43].

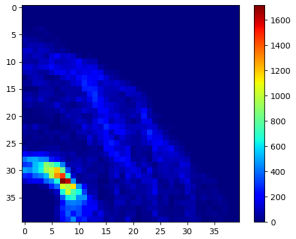
4.1.2. Relevant statistical physics literature. It should come as no surprise that a Markov chain as natural as the flip walk has been investigated before, especially given the interest in the self avoiding walk model. Indeed, the plane dual of the flip walk moves were applied to the study of self avoiding walks in $\mathbb{Z}^2$ with fixed endpoints (but *not* constrained to lie in a bounded region) in the BFACF algorithm [87, Section 6.7.1]. However, the state space of this walk is infinite, unlike our setting. It was proven that this walk has infinite exponential autocorrelation time [88]. Various efforts were made to improve the mixing time of the BFACF algorithm [87, Section 6.7.2], since physicists were interested in sampling from the stationary distribution in addition to observing the paths of the chain itself [86]. Additionally, Markov chains on self avoiding walks have been considered in constrained domains [87, p.69] just as in our setting, but it appears that little is known. In a somewhat different direction, and conditional on conjectures about the asymptotics of c_n, there are rapidly mixing Markov chains for uniformly sampling from unconstrained, fixed length self avoiding walks starting at the origin with free endpoint, see [82, 89].

Additionally, known bounds on self-avoiding walks provide estimates for the size of $P_2(L_n)$. For lower bounds, estimates on the number of self avoiding walks [73] can be used. For the upper bound, methods in section 5.1 of [24] can be used. Interestingly, [24] cites [11] as the inspiration for their method, and [11] was written to address the question of how many ways one could design districting plans for a grid-like state.

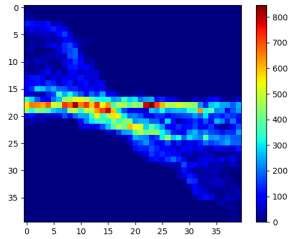
4.2. Experiments around mixing. Our goal in this section is to document experiments about MCMC methods built on top of the flip walk. These experiments were designed to investigate whether the chain was drawing samples from its stationary distribution. As the proposal distribution, we used the flip walk on the *simply-connected* elements of $P_2(L_n)$. We used a Metropolis score function $S((A,B)) = \lambda^{-|\text{cut}(A,B)|}$, so that the stationary distribution would be ν_λ. To tune λ around the critical fugacity $1/\mu$, we used the estimates $\mu \in [2.625622, 2.679193]$ and $\mu \approx 2.63815853031$ ([62, p. 10], which references [61, 81]). We also imposed balancedness constraints that prevented any partition from having more than $X\%$ of the total number of nodes beyond the number of nodes in a perfectly balanced partition, which we call an allowed population deviation (APD) of $X\%$.

These experiments are recorded in Figure 4.2 and Figure 4.3.

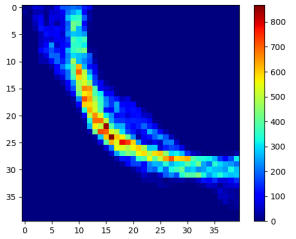
[9] We wish to thank a helpful conversation on MathOverflow for drawing our attention to this fact [4].



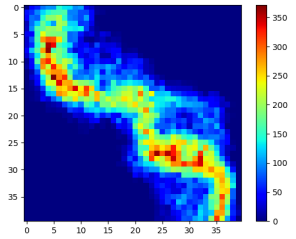
(a) *Very low fugacity, without tight population constraints* Population deviation 90%, $\lambda = 1/10$, Steps $= 5,072,065,569$. This quickly shrunk to a small bubble in the corner, around which the walk oscillated.



(b) *Very low fugacity, with tight population constraints* Population deviation 10%, $\lambda = 1/10 < 1/\mu$, Steps $= 2,905,381,156$ This quickly rotated from the diagonal to be horizontal, and oscillated around that.



(c) *Very low fugacity, with extremely tight population constraints* $APD = 1\%$, $\lambda = 1/10 < 1/\mu$, Steps $= 2,413,374,064$. The population restriction made it more difficult to escape the diagonal configuration.



(d) *Critical fugacity, with loose population constraints* $APD = 50\%$, $\lambda = 1/\mu$, Steps $= 1,405,646,093$. The walk oscillated around the diagonal.

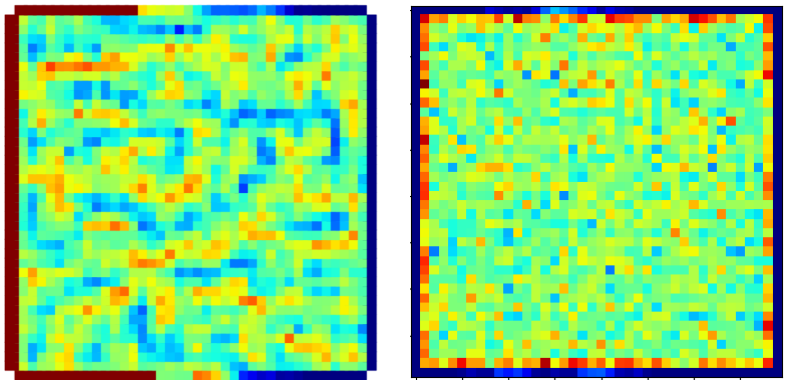
Figure 4.2: These examples start from an upper left to bottom right diagonal partition of a 40×40 grid graph. Each node kept track of the number of times it was flipped, and this number is reported and colored according to the key. Some interpretation of the history of the path revealed by the figures is provided. In all of these examples a symmetry argument demonstrates that the chain has evolved into some metastable region P_2.

4.2.1. Kansas State-Dual Graph. In this section, we repeat the experiments we performed on the grid graph the state dual graphs (§1.3) of Kansas, which was chosen because its state dual graph resembles the grid graph. For ease of visualization, we display the partitions and flip statistics on the underlying map rather than the state dual graph. The main features to observe are the slow mixing of the chain around the state space, and the phase transitions. See Figure 4.4.

4.3. Graph topology and phase transitions. So far we have discussed phase transitions of self-avoiding walks and connected 2-partitions in the grid graph, and remarked that the critical fugacity occurs at $1/\mu \approx .379$. However, there are other lattices, and for them the phase transitions occur at different values. Thus, one may wonder about the behavior of a self avoiding walk in a "Frankengraph" such as in Figure 4.5, which consists of triangles on the top, and squares on the bottom. In Figure 4.6 we show that we can find a value λ where the part of the partition boundary in the square grid acts super critically, and the part in the triangular acts subcritically.[10]

These experiments about phase transitions in geographic compactness scores fit in with other observations about compactness, namely that many features of the scores are not robust under changes of scale,

[10]Connected 2-partitions of the triangular lattice correspond to self avoiding walks in the dual lattice, which is a hexagonal lattice. Since [44] puts the connective constant of the hexagonal lattice at $\sqrt{2+\sqrt{2}}$, the phase transition for partitions of the triangular part occurs around .541. The authors wish to thank Sarah Cannon for discussions that clarified this behavior.



a) Average block b) Flips

Figure 4.3: Two measurements of a single run. $APD = 90\%$, $x = 1$, steps $= 194,390,536$, started from a vertical partition into red and blue blocks. a) Each node displays the color of its average block: more red if frequently in the red block, and more blue if frequently in the blue block. The left side, which started red, stays red through the entire run, while the right side stays blue. Considering the rotational symmetry and length of the run, this is a strong indication of a large mixing time. b) Each node records the number of times it flipped, with red indicating 100+, green/yellow indicating around $60-80$, and blue indicating < 20. One can see that most of the activity happens near the boundary of the square, and that the endpoints of the boundary of the partition barely move.

geometry or other implementation parameters [17, 18, 42]. Here we have raised an additional issue, which is that calibration of compactness score parameters in relation to the topology of the underlying graph can have dramatic affects on ensembles. These observations should be considered in light of [18], which analyzed the impact of decisions regarding the calculation of these compactness scores, and found that apparently small choices in implementation could have large effects.

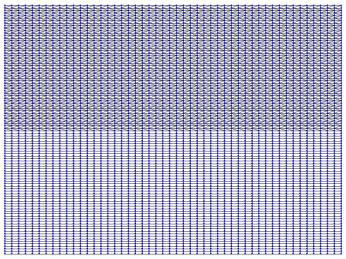
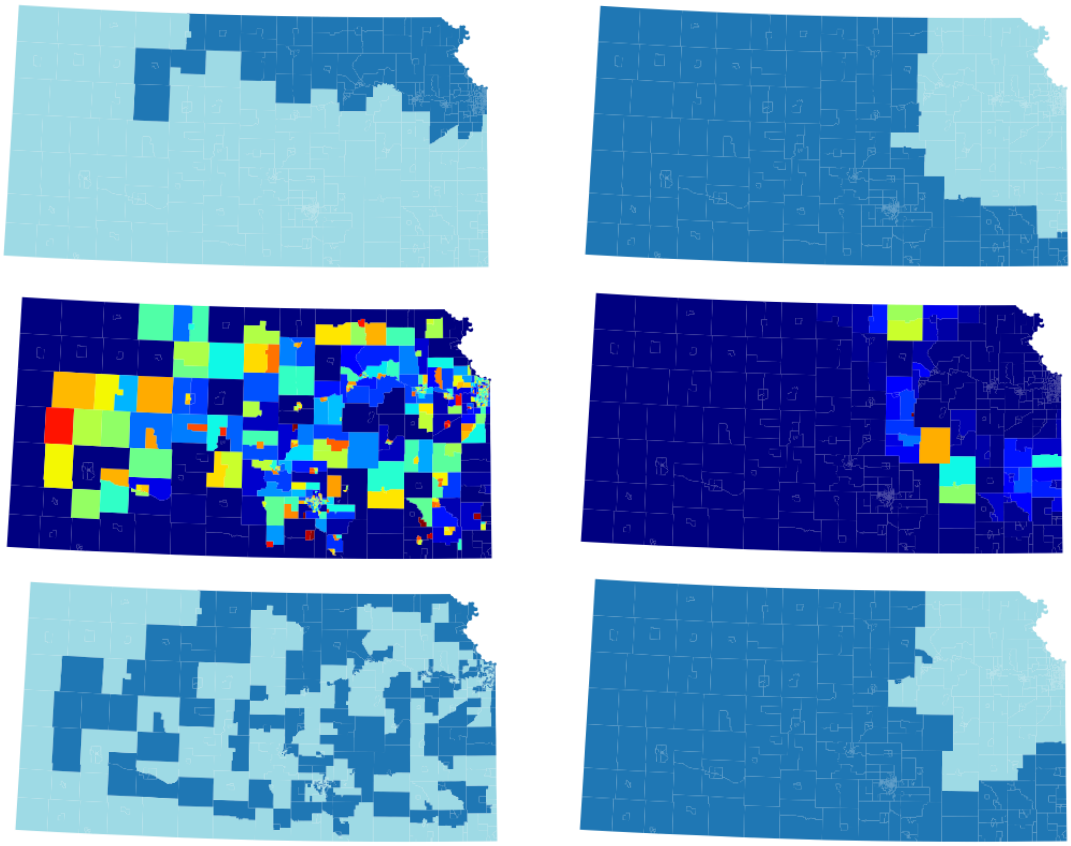


Figure 4.5: Frankengraph. The top is is a triangular lattice and bottom is a square lattice. Both pieces are 50×50. See Figure 4.6 for results of running the flip walk.

4.3.1. The choice of model graph. The example of the λ^{cut} distribution and the Frankengraph teaches us that the choice of graph used to discretize the same underlying geography can dramatically affect the distribution over partitions produced by a fixed algorithm. However, in those cases the geographic reasonableness of the distribution also changed dramatically, making it easy to classify a *single* partition as arising from one discretization or the other. In this section, we look at a different distribution over partitions of a rectangle, with the property that changing the discretization noticeably changes the distributions, but such that differences between the two distributions cannot be easily detected by observing natural geometric properties of individual plans.



a) $\lambda = 1$, $APD = 90\%$. Steps= $93,415,894$ b) $\lambda = .379$, $APD = 90\%$. Steps= $1,744,003,380$

Figure 4.4: Two runs of flip walk based $MCMC$ on the state dual graph of Kansas. Top: Starting plan. Middle: Counting Flips. Bottom: Ending plan.

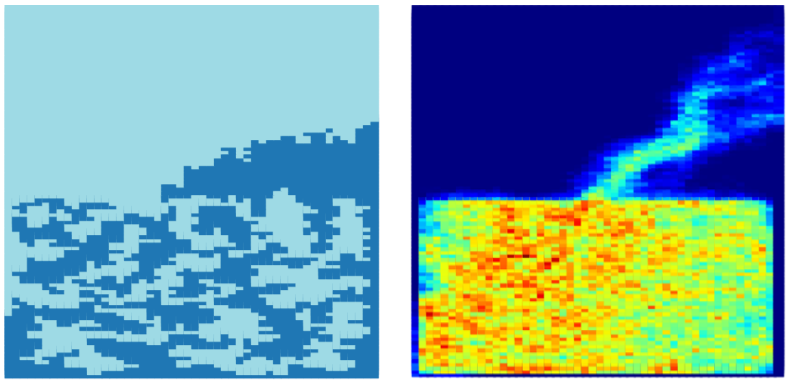


Figure 4.6: A flip walk based MCMC run with $\lambda = 1/2$, starting from the diagonal partition of the Franken-graph. 90% population constraint. $17,768,956,990$ steps.

We will now explain this other sampling algorithm, which underlies the method used in [36, 41]. Let UST be an algorithm that takes a graph G and returns a uniform spanning tree, for example using Wilson's algorithm [94]. Let MST refer to the minimum spanning tree obtained by picking iid Uniform($[0,1]$) edge weights. Let Tree(G) refer to either UST(G) or MST(G). Removing an edge e from Tree(G) gives a forest with two components, and hence an element of $P_2(G)$. If we repeat this algorithm, only selecting those edges that provide an ϵ-balanced connected 2-partition of G, then we obtain a distribution over $P_2^\epsilon(G)$. For both UST and MST, this distribution over partitions has some favorable properties, such as its concentration on partitions with small edge-cuts, which have made it appealing as a tool for sampling districting plans [36, Section 3.1.1]. [11] We will call this distribution UST-partition or MST-partition, or Tree-partition if we refer to either.

We construct a sequence of graphs from the 36×36 grid graph G by by triangulating a set of its faces in the following way. Fix some $w \in [0,3]$, and suppose that the nodes of the grid graph are labelled by (i,j), $i,j \in [0,35]$. For each $(x,y) \in V(G)$, if $12 \leq y \leq 20$ and $0 \leq x \leq 6w$ or $34 - 6w \leq x \leq 34$, add an edge $((x,y),(x+1,y+1))$ if x is even, and an edge $((x,y),(x+1,y-1))$ if x is odd. We call the resulting graph G_w, and w is referred to as the *width*. We think of partitions of G_w as modelling the same state, but with different choices regarding adjacency between geographic units. By changing the width, we can see a change in the shape of a typical Tree-partition. The results are displayed in Figure 4.7, where we use a number in $[0,3]$ to quantify the width of each half of the gate; the entire graph has dimensions 36×36, and the length of each half-gate is $width \times 6$. The effect is very clearly that closing the gap in the squeezes the boundary of the partitions in between the two halves of the gate.

In Figure 4.8 we show that MST-partitions and UST-partitions produce different partisan outlier measurements, despite the similarity between the description of the algorithms. In particular, if the underlying graph is is the one with width 2, and if the distribution chosen as a baseline for outlier analysis is MST-partition, then most of the time the seat share is 1, and many random samples from UST-partition would be considered extreme outliers. We discuss this further in §6.2.2.

We recall from §1.3 that in the analysis of a districting plan, one starts with a geographic entity, a U.S. state, that is broken into small geographic units. The choice of how the state is broken into small units determines an adjacency graph, and we modelled partitions of this state as connected partitions of that adjacency graph. If we intend to draw conclusions about the political geography of the underlying geographic entity, then one might hope that the impact of this modelling step is relatively tame. Figure 4.8 shows that this is not the case for the two examples above, potentially muddying the description of what outlier methods measure.

In political redistricting, there is occasionally reason to refine the basic geographic units, for example to balance population. A similar situation where changes to graph topology can occur is where choices have to be made about which units to connect across bodies of water, such as in [26], or when deciding between using rook or queen adjacency in the construction of the state dual graph.[12] There are also a variety of resolutions on which maps can be viewed, from counties to tracts to census blocks. All of this could potentially impact outlier analysis in a way similar to the Frankengraph example and Figure 4.7. Thus, someone sampling partitions of state dual graph to investigate the political geography of redistricting should keep in mind that they are making a potentially significant choice at the level of the choice of model graph. To comport with best practices in statistics [50], these decisions should be made as transparently and impartially as possible.

In many U.S. states, precincts, which are the atomic geographic units where electoral data is reported, are drawn at the discretion of the local municipalities. The examples in this section show that the people who choose the smaller geographic units potentially have a lot of control over the results reported by outlier methods. If outlier methods become standard, it would open the possibility of *metamandering* through the deliberate manipulation of these boundaries. No comprehensive analysis has been done to understand the impacts of these decisions.

[11] A fact partially reflected in [65, Corollary 2]; the connection being that the probability that a UST-partition $(A,B) \in P_2(G)$ is chosen is proportional to $T(A)T(B)cut(A,B)$, where $T(\_)$ counts the number of spanning trees, and based on this one can rearrange the asymptotics in Kenyon's paper to deduce that among the rectilinear partitions those with smaller perimeter are asymptotically preferred. However, rectilinear partitions are a set of extremely small measure in the UST-partition distribution, and so this explanation for concentration of UST-Partition on smaller fundamental cut-sets is only partial.

[12] That is, between declaring two units adjacent if they have a common edge, or if they have any common point.

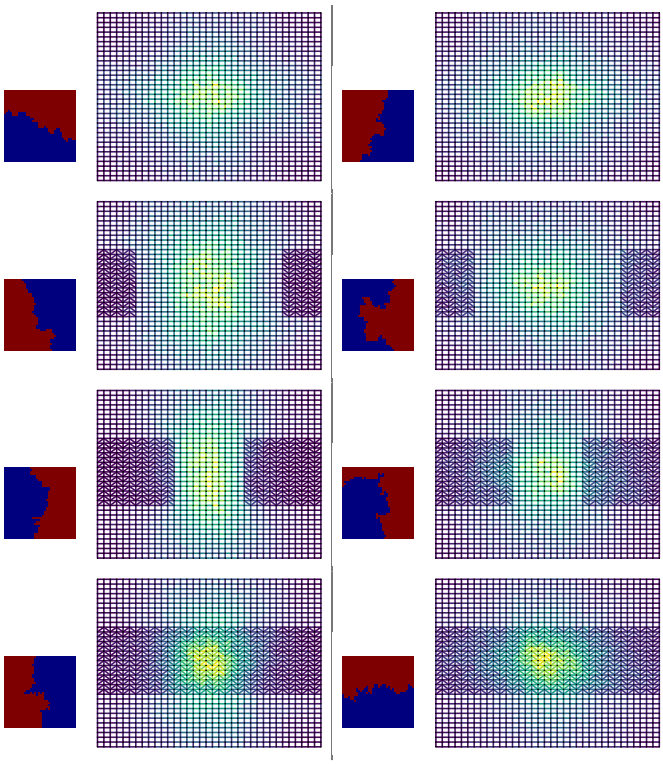


Figure 4.7: An edge is yellower if it is more frequently a cut-edge of sampled partition. The left hand diagrams were made using MST-partition, and the right hand side with UST-partition. Both use 1000 sampled plans, and plans are conditioned on having at most a 5% size deviation between the two blocks. Some sample partitions are displayed next to the cut-edge picture. The UST-partitions tend to have slightly longer boundaries on average.

5. Positive results. In this section, we provide several results regarding the tractability of sampling connected k-partitions and simple cycles. Most of these results will follow from the tractability of counting connected k-partitions on various families of graphs. Unlike many p-relations, connected 2-partitions and simple cycles do not appear to be encodeable in a self-reducible way (in the sense used in [63, 66]), meaning that the equivalence between counting and sampling requires variations on the ideas explained in [63].[13] However, we can use the chain of bigons construction to evaluate the marginal probabilities that self-reducibility would normally reduce to a counting problem. We can also directly modify the counting algorithms we find to compute the marginals. First, we recall how to use certain marginal probabilities to sample from a probability distribution over subsets of a given set.

5.1. Sampling from counting. The following algorithm is a standard part of the equivalence between counting and sampling, and is usually stated in the context of self-reducible structures. In Appendix B.1 we provide a proof of correctness.

Definition 5.1 (Marginal probabilities of distributions over subsets). *Let p be a probability distribution on $2^{[n]}$, the set of subsets of $[n] = \{1, 2, 3, \ldots, n\}$. Let S be a random variable distributed according to p. For a set $J \subseteq [i-1]$, define $p(i|J) = \mathbb{P}(i \in S | S \cap [i-1] = J)$; that is, the probability that S contains i, conditioned on containing J and being disjoint from $[i-1] \setminus J$. (As a convention, take $[0] = \emptyset$.)*

The use of the previous definition is in the following algorithm for sampling from a probability distribution

[13] Using techniques similar to [66] we can prove that at least one reasonable encoding of simple cycles is not self-reducible, unless P = NP. See Appendix B.5.

Width	MST	UST
0	1.346, 1.339	1.348, 1.273
1	1.045, 1.479	1.220, 1.314
2	1.003, 1.689	1.191, 1.425
3	1.072, 1.559	1.239, 1.379

Figure 4.8: 1000 samples, on a 36×36 node graph. Each half gate is $6 \times$ width vertices long, so width 3 represents the gate cutting the graph entirely in half. The numbers reported are obtained in the following way: some nodes are set to be 1 and others to 0. Each district majority votes to decide the "party" of the representative in that region, which is a number in $\{0, 1\}$. By summing the party across both districts, each plan is assigned a total party value in $\{0, 1, 2\}$. We have reported the mean total party value across 1000 trials, according to two different voting population distributions. The left hand numbers reported are based on a distribution of voters where the left 60 % of the nodes in the square are party 1, and the right hand numbers are based on a voter distribution where the bottom 60% are party 1. Since the voting data forces the total party value into $\{1, 2\}$, these are all Bernoulli random variables, and therefore the entire distribution can be read from the mean that we reported.

over subsets of a set, given access to the marginals of Definition 5.1.

Algorithm 5.1 `InductiveSampling`

Input: A probability distribution p on $2^{[n]}$ described via an oracle O that can compute $p(i|J)$ for any $i \in [n]$ and any $J \subseteq [n]$.
Output: A random element of $2^{[n]}$ distributed according to p.

1: Set $J_0 = \emptyset$
2: **for** $i \in [n]$ **do**
3: Use O to calculate $p(i|J_{i-1})$
4: With probability $p(i|J_{i-1})$, set $J_i = J_{i-1} \cup \{i\}$. Else, set $J_i = J_{i-1}$.
5: Return J_n.

The correctness of this algorithm (Appendix B.1) proves the following:

THEOREM 5.2. *Let $\mathscr{C}$ be a language encoding graphs (resp. node-weighted graphs), and suppose that there is a polynomial time Turing machine M (resp. M_B), that on input $G \in \mathscr{C}$, $J, J' \subseteq E(G)$, computes the number of simple cycles containing J and disjoint from J' (resp. the number of balanced connected 2-partitions whose cut set contains J and is disjoint from J'). Then there is a polynomial time probabilistic Turing machine that uniformly samples from $SC(G)$ (resp. uniformly samples from $P_2^0(G)$) for $G \in \mathscr{C}$.*

5.2. Remarks on algorithmic meta-theorems. The next sections will be concerned with computing the marginals that are necessary for Theorem 5.2. This will be done by showing that we can solve certain counting problems. The tractability of the counting problems on graphs of bounded treewidth will follow from extensions of Courcelle's theorem, such as those in [14]. In particular, the cut edges of a connected k-partition can be expressed in MSO_2 (see §5.4.2), and similarly the cut edges of a balanced connected k-partition can be expressed in EMS, as defined in [14].[14] The constants in these meta-theorems are too large to be practically useful, as the automata on which they are based grows in size like a tower of exponentials in the size of the formula, and there are no general tricks to avoid this [48] . Therefore, although we appeal to these meta-theorems to conclude complexity theory statements, we emphasize practical approaches to solving these counting and sampling problems on series-parallel graphs, which give some directions for practical

[14]For background on second order logic, the reader is referred to [46, Chapter 7]; for background on these meta-theorems and MSO_2, the reader is referred to [14]. A brief summary of the meta-theorem that we use is given in §5.4.1.

implementations on wider classes of graphs. For example, forthcoming work [91] extends the ideas applied to series-parallel graphs to arrive at a reasonably implementable algorithm for counting and sampling simple cycles fixed-parameter tractably in the treewidth.

5.3. Simple cycles.

DEFINITION 5.3 (Marginal graph). *Given a graph $G = (V, E)$ and $J, J' \subseteq E$, let $G_{J,J'}(d)$ denote the graph where the edges in J' are deleted, and the edges in J are replaced by a chain of d bigons.*

The next lemma shows that for sufficiently large d, the number of simple cycles in G containing J and disjoint from J' can be inferred from $|SC(G_{J,J'}(d))|$, by using division with remainder and the same exponential growth rate comparisons that drove the intractability results:

DEFINITION 5.4. *If $\mathscr{C}$ is some language encoding graphs, and $k : \mathscr{C} \to \mathbb{N}$ some function, then we call k a parametrized language of graphs.*

LEMMA 5.5. *Let $k : \mathscr{C} \to \mathbb{N}$ be a parametrized language of graphs. Suppose that there is a polynomial p and a computable function f and a Turing machine M that can calculate $|SC(G_{J,J'}(d))|$ in time $f(k(G_{J,J'}(d)))p(|G|, d)$ for all $G \in \mathscr{C}$ and $d \geq 1$ and for any $J, J' \subseteq E(G)$. Then, there is a polynomial q and a TM which calculates $b_{J,J'} := |\{T \in SC(G) : J \subseteq T, J' \cap T = \emptyset\}|$ in time $O(f(k(G_{J,J'}(36n^4)))q(|G|))$ for all $G \in \mathscr{C}$ and $J, J \subseteq E(G)$.*

Proof. If $|J| = 0$, then $|SC(G_{J,J'}(d))| = b_{J,J'}$. We assume that $|J| \geq 1$, and let $n = |G|$. Observe, in the manner of Proposition 2.13, that

$$|SC(G_{J,J'}(d))| = \sum_{k=0}^{|J|} 2^{dk} |\{X \in SC(G) : X \cap J' = \emptyset, |X \cap J| = k\}| + d|J.|$$

We define $a_d = |SC(G_{J,J'}(d))|$ and the remainder term

$$R_d = \sum_{k=0}^{|J|-1} 2^{dk} |\{X \in SC(G) : X \cap J' = \emptyset, |X \cap J| = k\}| + d|J|.$$

R_d is bounded above by $2^{d(|J|-1)}2^{n^2} + d|J| \leq dn^2 2^{d(|J|-1)+n^2}$. By Lemma 2.10, for $d = 4\lceil(n^2 + \log(n^2) + 1)^2\rceil$, $2^{d|J|} > dn^2 2^{d(|J|-1)+n^2}$, and hence for such d, we have $a = 2^{d|J|} b_{J,J'} + R_d$, where $2^{d|J|} > R_d$. Since each term in that expression is an integer, R_d is the remainder of dividing a by $2^{d|J|}$, and $b_{J,J'}$ is the quotient. This division with remainder can be performed in $O((\log(|SC(G_{J,J'}(d))|) \log(2^{d|J|})))$ time [84, Theorem 3.3], which is polynomial in $(|G|, d)$. Since $d = 4\lceil(n^2 + \log(n^2) + 1)^2\rceil \leq 36n^4$, the result follows. ∎

We briefly recall a definition of treewidth:

DEFINITION 5.6 (k-trees, partial k-trees and treewidth). *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. The treewidth of a graph G is the smallest k such that G is a partial k-tree.*

The operation $G \to G_{J,J'}(d)$ preserves the class of series-parallel graphs, and in addition, does not increase the treewidth of any graph with treewidth ≥ 2. There are polynomial time dynamic programming algorithms for counting the number of simple cycles of a series-parallel graph,[15] and it follows from Courcelle's theorem that counting the number of simple cycles is fixed-parameter tractable in the treewidth.[16] Thus, from Lemma 5.5 and Theorem 5.2 we have the following:

THEOREM 5.7. *The problem of uniformly sampling simple cycles is* FPT *in the treewidth.*

[15] See Appendix B.3

[16] This follows from Proposition 5.11 and Theorem 6.56 in [34]. We are grateful to Mamadou Moustapha Kanté for pointing this out on Stack Exchange [5]. There is an explicit MSO_2 formula for simple cycles at the same link.

Since the treewidth of a plane dual changes by at most one [23,68], we obtain a similar result for sampling connected 2-partitions. However, in the next section we will show how to apply Courcelle's theorem directly to show that sampling connected k-partitions is FPT in the treewidth.

The treewidth of a typical state dual graph used for redistricting (§1.3) is on the order of 40 to 60; although this shows that treewidth is not a useful parameter for state dual graphs, this does not rule out the possibility that other parameters can make sampling from $P_2(G)$ tractable when G is a state dual graph. We discuss this further in §6.3.

With a little more work, one can show that many other distributions over simple cycles besides the uniform distribution can be efficiently sampled from, at least on graphs with treewidth ≤ 2. We now introduce a definition to describe these distributions.

Definition 5.8 (Edge weight probability). *Let G be a graph, and $c : E(G) \to \mathbb{Q}_{\geq 0}$ some weight function. Let N_c denote the measure on simple cycles that gives weight $N_c(C) = \prod_{e \in C} c(e)$ to each simple cycle C, and let ν_c denote the probability distribution obtained by normalizing N_c.*

Theorem 5.9. *Sampling from ν_c is polynomial-time solvable on the class of graphs of treewidth ≤ 2.*

Proof. See Lemma B.14, which shows how to directly compute the required marginal probabilities for sampling from ν_c via Algorithm 5.1, without using Lemma 5.5. ∎

The space of distributions on $SC(G)$ is *far* larger than the distributions described by ν_c. We mention several other tractable distributions on $SC(G)$ and $P_2(G)$ in §5.6.

5.4. Connected k-partitions. First, we will briefly review MSO_2 and the counting meta-theorem in §5.4.1. Next, in §5.4.2, we will describe an MSO_2 formula that defines connected k-partitions. Finally, we will tie these together to prove the following:

Theorem 5.1. *Uniformly sampling from $P_k(G)$ is* FPT *in the treewidth.*

5.4.1. Counting solutions to MSO_2 formulas. We mostly follow [14]. We consider a relational vocabulary $\mathcal{R} = (V, E, J, J', \mathrm{inc})$, where V, E, J, J' are unary relations, and inc is a binary relation. Additionally, we consider a set of formulas $\Gamma = \{\forall x V(x) \vee E(x), \forall x V(x) \leftrightarrow \neg E(x), \forall x \forall y \mathrm{inc}(x, y) \to V(x) \wedge E(y), \forall x J(x) \vee J'(x) \to E(x)\}$. $\mathrm{Mod}(\Gamma)$ denotes the set of models of Γ. Given a model of Γ with universe A, A is partitioned by the two sets defined by V and E, which we refer to as V and E, by abuse of notation. We interpret V as the set of vertices, E as the set of edges, J and J' as two collections of edges, and inc as the incidence relation. That is, $\mathrm{inc}(v, e)$ is interpreted as meaning that vertex v is incident to edge e. Thus, a model for Γ is a graph along with two sets of edges.

We denote by MSO_2 the second order logic with signature $\mathcal{R}$ that allows only unary relational variables. Given a formula $\Phi(X)$ in MSO_2 with a free variable X, the enumeration problem for Φ is that of computing $|\{X : G \models \Phi(X)\}|$ for any given $G \in \mathrm{Mod}(\Gamma)$.

Then, we have:

Theorem 5.2. *[14, Theorem 5.7] For each MSO_2 formula $\Phi(X)$, and for each class K of graphs with universally bounded treewidth, the enumeration problem for Φ can be solved in $O(|G| \log(|G|))$ time if G is given with a tree-decomposition.*

For the purposes of obtaining an MSO_2 formula, it is convenient to represent a connected k-partition by the *complement* of the cut-set, similarly to Appendix A.3.

Definition 5.10 (Connected partitions as edge sets). *Given an (unordered) k-partition P, define $F(P) = \mathrm{cut}(P)^c$. Let $\mathrm{Flats}_k(G)$ be the set $\{F(P) : P \in P_k(G)\}$. For any $J', J \subseteq E$, define $F_{J,J'}(G) \subseteq \mathrm{Flats}_k(G)$ as those $Q \in \mathrm{Flats}_k(G)$ with $J \subseteq Q$ and $Q \cap J' = \emptyset$.*

Since a connected k-partition is determined by its cut set (Proposition A.10), it follows that $F : \mathscr{P}_k(G) \to \mathrm{Flats}_k(G)$ is a bijection. We are describing (unordered) connected partitions as flats in the graphic matroid, hence the notation "F" and "Flats_k."

5.4.2. MSO_2 formula for edge sets in $\mathrm{Flats}_k(G)$. Let $G = (V, E)$ be a graph. For $X \subseteq E$ we will build up to an MSO_2 formula that checks if $X \in \mathrm{Flats}_k(G)$. Our building blocks are inspired by the examples in [35, Chapter 7]. First, we define a formula that checks if a set of nodes, Y, is contained in $G[X]$:

$$\text{In}(Y, X) = \forall_{v \in Y} \exists_{e \in X} \text{inc}(v, e).$$

Given two sets of vertices, U and W, we define a formula that checks if there is an edge in X connecting a node in U to a node in W:

$$\text{Bridge}(U, W, X) = \exists_{u \in U, w \in W, e \in E} \text{inc}(u, e) \wedge \text{inc}(w, e).$$

Next we define a formula that checks if $G[X]$ is connected, by checking whether there are any non-trivial 2-partitions of $V(G[X])$ with no edges in X between the different blocks.

$$\text{connE}(X) := \forall_{Y \subseteq V} [\text{In}(Y, X) \wedge Y \neq \emptyset \wedge [\exists_{U \subseteq V} \text{In}(U, X) \wedge U \cap Y = \emptyset \wedge U \neq \emptyset \wedge U \cup Y = V]] \rightarrow \text{Bridge}(Y, U, X).$$

We next define a formula that checks if an edge has both endpoints in the nodes of a subgraph induced by a set of edges Y:

$$\text{ep}(e, Y) = \forall_{v \in V} \text{inc}(e, v) \rightarrow (\exists_{e' \in Y} \text{inc}(e', v))$$

Finally, for each $k \geq 1$, we define a formula that takes a collection of edges, X, and checks whether it is in $\text{Flats}_k(G)$. This is accomplished by checking that every node of G is incident to some edge in X and that X is a union of k sets of edges, each of which induces a connected subgraph and so that any edge with both endpoints in one of those connected subgraphs is in X.

$$F'_k(X) = \text{In}(V, X) \wedge (\exists_{X_1, \ldots, X_k \subseteq E} (X = \bigcup X_i \wedge \bigwedge_i connE(X_i) \wedge (\forall_{e \in E} \bigwedge_i (\text{ep}(e, X_i) \rightarrow e \in X_i)))$$

Recall that we considered J and J' to be part of the relational structure $\mathcal{R}$, so we can define the MSO_2 formula whose solution sets are the members of $F_{J,J'}$:

$$F_k(X) = F'_k(X) \wedge (J \subseteq X) \wedge (J' \cap X = \emptyset) \tag{5.1}$$

LEMMA 5.3. *Let A be a model of Γ, i.e. a graph $G = (V, E)$ with vertex-edge incidence matrix given by* inc *and two distinguished subsets of edges, J and J'. $F_k(X)$ is true in A if and only if $X = F(P)$ for some connected k-partition P of G and $X \cap J' = \emptyset$ and $J \subseteq X$.*

We now prove Theorem 5.1.

Proof. Let K be a class of graphs with universally bounded treewidth. For any $G \in K$ and $J, J' \subseteq E(G)$, by Theorem 5.2 and Lemma 5.3, we can count $|F_{J,J'}(G)|$ in time $O(|G| \log(|G|)$ with constant dependent only on the bound on the treewidth and the formula F_k. The conditions for running Algorithm 5.1 are satisfied. ∎

REMARK 5.11. *It is easy to add a relational formula (see [14]) to Equation (5.1) that restricts our count to only balanced connected k-partitions. In particular, the balanced connected k-partition problem is in* extended *monadic second order logic (EMS). From this it should follow that so the counting and sampling problems are* XP *in the treewidth. However, as noted at this Stack Exchange question [9], the corresponding meta-theorem appears to be missing from the literature.*

5.5. Balanced 2-partitions. We mentioned in §2.4 that [45] proved that determining if a graph has a balanced connected 2-partition is NP-hard. That paper also describes a dynamic programming algorithm that determines if a series-parallel graph has a balanced connected 2-partition. This dynamic programming algorithm can be modified to produce an algorithm for *counting* the number of balanced connected 2-partitions of a given node-weighted series-parallel graph G in time polynomial in G and pseudopolynomial in the weights. We present the details of this algorithm in Appendix B.4. To turn such a counting algorithm into an algorithm for calculating the marginals necessary for Theorem 5.2, we proceed along similar lines as in the simple cycle case.

DEFINITION 5.12 ($W^{J,J'}(G,w)(d)$). *Let (G,w) be a node weighted graph, and let $J, J' \subseteq E(G)$. Define $G^{J,J'}$ by replacing edges in J with the doubled d-star gadgets from Definition 3.8 and contracting the edges in J', deleting any self loops that arise in this way. Assign the "new nodes" of $D_d(e)$ weight 0 for each $e \in J$, and the old nodes the same weight as they had in G. The resulting node-weighted graph is denoted $W^{J,J'}(G,w)(d)$.*

We now show that the marginals necessary for Algorithm 5.1 can be computed from $|P_2^0(W^{J,J'}(G,w)(d))|$ (with notation as in §2.4) using division with remainder and the exponential growth rate calculations that drove the intractability result in §2.4:

PROPOSITION 5.13. *Let (G,w) be a weighted graph. Then:*

$$|P_2^0(W^{J,J'}(G,w)(d))| = 2^{d|J|}|\{X \in P_2^0(G,w) : J \subseteq \mathrm{cut}(X), \mathrm{cut}(X) \cap J' = \emptyset\}| + R_d, \tag{5.2}$$

where R_d is a non-negative integer with:

$$R_d \leq 2^{n^2} 2^{d(|J|-1)}. \tag{5.3}$$

Proof. Let G/J' denote the quotient graph obtained by identifying $u, v \in V(G)$ if $\{u,v\} \in J'$. First, we decompose

$$P_2(G/J') = \bigcup_{k=0}^{|J|} \{(A,B) \in P_2(G/J') : |\mathrm{cut}(A,B) \cap J| = k\}. \tag{5.4}$$

We define $R_J : P_2(G^{J,J'}(d)) \to P_2(G/J')$ as R_d is in Definition 3.9 by forgetting the assignment of new nodes, We pull back Equation (5.4) along R_J to obtain:

$$P_2(G^{J,J'}(d)) = \bigcup_{k=0}^{|J|} R_J^{-1}(\{(A,B) \in P_2(G/J') : |\mathrm{cut}(A,B) \cap J| = k\}).$$

The map $\phi^* : P_2(G/J') \to P_2(G)$ defined by $\phi^*((A,B)) = (\phi^{-1}(A), \phi^{-1}(B))$ is an injection, and the image is $\{(A,B) \in P_2(G) : \mathrm{cut}(A,B) \cap J' = \emptyset\}$.

Hence we have a *partition* of $P_2(G^{J,J'}(d))$,

$$P_2(G^{J,J'}(d)) = \bigcup_{k=0}^{|J|} (\phi^*_{J'} \circ R_J)^{-1}(\{(A,B) \in P_2(G) : \mathrm{cut}(A,B) \cap J' = \emptyset, |\mathrm{cut}(A,B) \cap J| = k\}).$$

So far we have decomposed the set of partitions of $G^{J,J'}(d)$. Next, we compute the 0-balanced partitions in each block of that decomposition. The elements of $(\phi^*_{J'} \circ R_J)^{-1}(\{(A,B) \in P_2(G) : \mathrm{cut}(A,B) \cap J' = \emptyset, |\mathrm{cut}(A,B) \cap J| = k\})$ are obtained by extending a partition in $\{(A,B) \in P_2(G) : \mathrm{cut}(A,B) \cap J' = \emptyset, |\mathrm{cut}(A,B) \cap J| = k\}$ onto the new nodes. Since each new node has weight 0, it is impossible to assign new nodes in such a way as to make unbalanced partitions of G balanced.

The balanced partitions that have J contained in the cut have exactly $2^{d|J|}$ balanced extensions each. This proves Equation (5.2). We are left to show the upper bound of Equation (5.3) for the reamining partitions, namely :

$$\mathrm{Rem}_d = (\bigcup_{k=0}^{|J|-1} (\phi^*_{J'} \circ R_J)^{-1}(\{(A,B) \in P_2(G) : \mathrm{cut}(A,B) \cap J' = \emptyset, |\mathrm{cut}(A,B) \cap J| = k\})) \cap P_2^0(W^{J,J'}(G)(d))$$

We have that $R_d = |\mathrm{Rem}_d|$. Suppose that X is some balanced partition of G, with $|\mathrm{cut}(X) \cap J| \leq |J|-1$. The number of ways to extend X to the new nodes and get a balanced partition is at most $2^{d(|J|-1)}$. Since $|P_2(G)| \leq 2^{n^2}$, this provides the upper bound on the remainder term. ∎

PROPOSITION 5.14. *Let $\mathscr{C}$ be some class of graphs that is closed under the operation $G \to G^{J,J'}(d)$ of Definition 5.12, for all $d \geq 1$. Let p be a polynomial. Suppose that M is a Turing machine which can compute $|P_2^0(G)|$ on all weighted graphs (G, w) where $G \in \mathscr{C}$ and $w : V(G) \to \{0, 1, \dots, \}$, in time bounded by $p(|G|, w(G))$. Then there is a polynomial time probabilistic Turing machine that uniformly samples from $P_2^0(G, w)$ in time polynomial in $(|G|, w(G))$ for all $G \in \mathscr{C}$.*

Proof. Due to Algorithm 5.1, to sample in polynomial time it suffices to be able to compute $a_{J,J'} := |\{X \in P_2^0(G, w) : |\text{cut}(X) \cap J| = |J|, \text{cut}(X) \cap J' = \emptyset\}|$ in polynomial time for any given $J, J' \subseteq E(G)$. We will do this by from computing $|P_2^0(W^{J,J'}(G, w)(d))|$ at a value of d which is polynomially large in $|G|$.

If $d = n^2 + 1$, then $2^{d|J|} > 2^{n^2} 2^{d(|J|-1)}$. Now, given $N_d = |P_2^0(W^{J,J'}(G, w)(d))|$, from Proposition 5.13 we know that we can write $N_d = a_{J,J'} 2^{d|J|} + R_d$. Since we can efficiently compute $2^{d|J|}$ and N_d in time polynomial in $(|G|, w(G))$, since we fixed $d = n^2 + 1$, by division with remainder we can compute $a_{J,J'}$ in time polynomial in $(|G|, w(G))$. Thus, we have calculated the marginal that we need for sampling. ∎

THEOREM 5.15. *There is an algorithm for uniformly sampling from the balanced partitions of a node weighted* series-parallel *graph (G, w), which runs in time polynomial in $(|G|, w(G))$.*

Proof. This follows from Proposition 5.14 and the dynamic program for counting balanced partitions on series-parallel graphs presented in Appendix B.4, since the class of node weighted series-parallel graphs is closed under the operation $G \to G^{J,J'}(d)$ for all $d \geq 1$; this is because series-parallel graphs are closed under replacing edges by doubled d-trees, and under edge contractions (provided we eliminate self loops). ∎

REMARK 5.16. *It may be possible to extend this to an XP in treewidth algorithm for sampling balanced k-partitions, using similar ideas as well as those mentioned in the conclusion of [60].*

5.6. Other families of distributions over partitions. We conclude this section by pointing out that *many* distributions on $P_k(G)$ and $P_k^0(G)$ are tractable to sample. A general strategy for building k-partitions of G is to contract G in some random way onto a simpler graph, G', and then pull back k-partitions from the simpler graph. The following lemma shows that one can pull back connected k-partitions along quotient maps obtained by contracting connected partitions:

LEMMA 5.17. *Let G be a graph, and let $\phi : G \to G/R$ be a graph quotient map, where R is an equivalence relation on the nodes such that the equivalence classes of R induce connected subgraphs. Then for any $(A_1, \dots, A_k) \in P_k(G/R)$, $(\phi^{-1}(A_1), \dots, \phi^{-1}(A_k)) \in P_k(G)$. Moreover, if w is a node weight function on G, then if we assign each equivalence class of G/R the total weight of all its elements, ϕ^{-1} preserves the weight of blocks, and thus also pulls back balanced partitions.*

This lemma can be used to give a recipe for chaining together random partitions into of G with many blocks into an algorithm for obtaining random partitions into k blocks. For example, at each stage one can take R to be an equivalence relation induced by random sets of edges, such as the edges of a random matching, or the monochromatic edges in a sample from distribution over colorings, or a random forest, as we did in §4.3.1. One could also imagine finding random quotients onto graphs of smaller treewidth, and then using the sampling algorithms from the previous sections. Additionally, it is known [38,39, Theorem 11] that plane graphs can be *contracted* onto partially triangulated grid graphs with similar treewidth, which suggests that understanding the connected k-partition sampling problem for partially triangulated grid graphs is an open problem with important implications with sampling connected partitions of state dual graphs.

Another means of producing connected 2-partitions is via min cuts, since min cuts are always connected. There are polynomial time algorithms for uniformly sampling min s, t-cuts genus g graphs given in [27]. On general graphs, one can also sample min-cuts in a way that is fixed parameter tractable in the size of the min-cut [20], but the running time of this algorithm is practical only for very small min-cut sizes.

We emphasize that even though these distributions can be efficiently sampled from, it is not clear how to characterize their properties in terms of interpretable features of districting plans.As we discussed in §4.3.1, the properties of these distributions as distributions over partitions of the underlying geography may vary significantly with the discretization, and in a given application one needs to decide if this is acceptable.

6. Conclusions.

6.1. Broad overview of paper. We motivated this paper by discussing attempts at characterizing outlier redistricting plans through ensemble methods (§1.3). In practice this is applied by picking some statistic of interest and comparing the distribution of this statistic over sample maps to those of a proposed plan. The theoretical and experimental results in this paper indicate that additional considerations are needed before we can put full trust into statistical outlier analysis of gerrymandering. In particular:

- The flip walk proposal distribution used in practice is likely not rapidly mixing (§3 and §4.2), even on classes of graphs where sampling is tractable (§5).
- The complexity results (Theorem 2.43, Proposition 2.20 and Theorem 2.68) show that for many classes of graphs and distributions, there are likely to be no efficient replacements for the flip walk when it comes to sampling from certain prescribed distributions.
- Even if it were possible to sample from an explicitly designed distribution, that distribution may undergo phase changes in its qualitative behavior if the description is slightly modified (§4.3). Along similar lines, a fixed algorithm can produce dramatically different distributions of partitions over the underlying geography if different choices are made regarding the choice of model graph (Figure 4.5 and §4.3.1).

Altogether, these observations imply that inferential conclusions *may* not be robust to small changes in the set up.

6.2. Future directions. We now describe some directions that the computational redistricting community can go in to address these challenges.

6.2.1. Other applications to redistricting. There are local measurements of gerrymandering that do not require guarantees about sampling. One example [31, 80] is supported by a rigorous theory for a particular meaning of gerrymandering regarding "carefully crafted" plans. It remains an important question to investigate the extent to which the decisions we highlighted in §4.3 and §4.3.1 affect the interpretation of these tests.

Additionally, it is possible to produce a large ensemble of partitions using the flip walk [16], the means described in §5.6 and other methods such as [30, 33, 41]. Despite the difficulties inherent in characterizing the statistical behavior of these ensembles, a large collection of plans may be amenable for other species of analysis. For example, questions about the existence of plans meeting certain criteria can sometimes be answered by identifying demonstration plans, rough characteristics of trade-offs between criteria can perhaps be calculated, and certain implications of proposed legislation can be evaluated [36].

6.2.2. The extreme outlier hypothesis. Currently a consensus is developing that the notion of an *extreme* partisan outlier is robust between different sampling methods. For example, handful of recent court decisions reference favorably the outcome of such outlier analysis [40, 79, 95]. The consensus asserts that different ensemble methods are measuring a consistent and interpretable feature of the political data because those methods used in practice seem to detect the same extreme partisan outliers.

The hypothesis that there is a consistent and robust notion of an extreme outlier, which we will call the "extreme outlier hypothesis" (EOH), is likely to be a rich source of challenges and questions about ensemble based redistricting. This hypothesis was partially explored in [36], where it was found that certain changes in the sampling algorithm had little effect on the tails of some chosen statistics, but that other changes caused certain tails to become exaggerated. However, the changes that exaggerated the tails resulted from interpreting different legal constraints for permissible maps; for example, section 3.4 of [36] shows some examples of how adherence to the voting rights act can have dramatic impacts on the distribution of partisan scores.

It is easy to fabricate distributions where any specific plan appears to be an outlier, but such fabricated distributions may not reflect principles important in real world redistricting. To be useful, EOH must incorporate redistricting principles and not just mathematical abstractions. For example, in §4.3 we obtained two distributions described by similar parameters, which nevertheless find different extreme outliers. However, this does not falsify the EOH, because supercritical partitions are not representative of legal maps. Therefore, one could reasonably object to the baseline provided by the supercritical distribution, while accepting the usefulness of the baseline provided by the subcritical distribution. On the other hand, as we saw in §4.3.1, certain changes to the underlying discretization can shift its partisan properties without changing a typical

district's geometry, a more subtle change that may or may not have bearing on the EOH in practice. Of our experiments, the most challenging for the EOH is the significant difference between the UST-partition and MST-partition in the third row of Figure 4.8.

A reasonable formulation of EOH hypothesizes that it is implausibly difficult to fabricate distributions over districting plans that are defensible as a baseline for redistricting, even under scrutiny by adversarial experts, but which report different *extreme* outliers. To falsify EOH, it would *suffice* to find reasonable operationalizations of the same set of legal requirements into different sampling algorithms, which nonetheless report different extreme outliers. The problem of establishing precise guidelines for what constitutes a representative distribution over districting plans is understudied and critical to understanding the EOH.

6.2.3. Pragmatism and distributional design. A pragmatic resolution to the hard questions raised by the EOH may be to pick a handful of sampling algorithms that will be consistently recognized a baseline. Some states already incorporate restrictions on the use of partisan data in the drawing of districting plans, and ensemble based methods could be one additional way for those states to operationalize these intentions. However, there is likely not a single collection of distributions that will suit every geography and political culture. Beyond purely mathematical analysis, one route to finding suitable distributions is through empirical analysis and successive refinements under real world conditions.

Although the procedure for determining which distributions to set as baselines is politically fraught, understanding the sampling algorithms to promote informed decisions is a hard scientific problem. In addition to characterizing the distributions generated by these algorithms, an analysis of algorithmic reliability, as in §3 and §4.3, is important for developing standards that constrain data dredging and post-hoc analysis. Such reliability considerations are also important for reproducibility. Similarly, we should understand the dependence of partition sampling algorithms on the discretization, to constrain what we called *metamandering* in §4.3.1.

Using sampling algorithms to detect partisan intent and using sampling algorithms to constrain possibilities are two separate tasks. A clear danger in both approaches is that distributions could be chosen in a way that creates unnecessary constraints or undesirable bias. Indeed, *if* the EOH fails, then giving someone the power to choose a baseline distribution creates the opportunity for subtler manipulation of voting outcomes. Along with exploring the EOH, developing empirically motivated partition sampling algorithms and understanding their trade-offs is a key direction for future research in this area.

6.3. Open questions. We summarize a handful of remaining questions about sampling connected partitions:

- What sort of ground truth models could be useful for assessing the accuracy of outlier methods? Of the districting plans historically or presently used, what proportion of them are flagged as outliers by suggested methodology? Does it correlate with other evidence for gerrymandering? Is the flagging consistent between methods?
- Can the intractability results be unified and strengthened? It seems unlikely that Theorem 2.25 is optimal.
- Can a general and practically useful sufficient for the existence of a bottleneck in the flip walk be extracted from the examples in §3?
- Besides treewidth, are there other graph parameters that make uniformly sampling from $P_2(G)$ tractable?
- All of our intractability results relied on reductions from Hamiltonian cycle, by proving that any algorithm sampling from certain distributions can be modified to put large mass on the longer simple cycles of a graph. However, the partitions that are of interest to redistricting tend to have relatively short boundaries on the order of $\Theta(\sqrt{|V|})$, rather than $\Theta(|V|)$. As mentioned in §5.6, it is possible to sample from the set of min-cuts, FPT in the size of the min-cut. However, this is a different regime from sampling from the cuts of size $\Theta(\sqrt{|V|})$ for the graphs that arise as state dual graphs (§1.3). Are there approaches to proving tractability or intractability of sampling such medium length cycles?
- Is it possible to uniformly sample from $P_2(L_n)$, where L_n is the $n \times n$ grid graph from §4.1? What about if we consider the class of graphs obtained from L_n by adding some diagonal edges as in Figure 4.7, or partially triangulated grid graphs [38].
- Are there families of graph with unbounded treewidth where ν_λ sampling $P_2(G)$ is tractable?
- Statistical evidence, included repeating the tests in [64] as well as the flip pictures in §4.2, suggests that the ν_λ Metropolis-Hastings weighted flip walk Markov chain on $P_2(L_n)$ mixes rapidly only at the critical

value $\lambda = 1/\mu$. Is this true? What is the dependence on the population balance restriction?

- Which distributions over $P_k(G)$ can we efficiently sample from? Which of these distributions is robust to changes in the discretization?
- Recalling the motivation (§1.3) and problems raised by discretization (§4.3.1), one may be inspired to abandon the discrete model and directly sample from partitions of the underlying geography. There are many sampling algorithms one can investigate here, such as some derived from Schramm-Loewner evolution, random lines or polynomial curves. What favorable or unfavorable properties do these sampling algorithms have?
- Although there are many plans, many of them are similar in shape. This may lead one to guess that there is a small collection of plans that are near to every other plan; i.e. that there is an epsilon net in the space of reasonable plans. For low dimensional shapes, it is reasonable to find an epsilon net, but for shapes of dimension d, the number of points needed to form an epsilon net grows roughly like $(1/\epsilon)^d$. It would be interesting to determine whether or not the space of reasonable plans was high or low dimensional from this point of view. The authors conjecture that this space will behave as if extremely high dimensional, but if there are ways to constrain it to be low dimensional, then the potential existence of a computable ϵ-net opens another way to discuss typicality while being distribution agnostic, for example through the analysis of Pareto fronts between measurements.

Acknowledgements. We want to thank the following people for their patience, enthusiasm, eagerness to share knowledge, insightful questions and helpful discussions: Hugo Akitaya, Eric Bach, Assaf Bar-Natan, Jin-Yi Cai, Sarah Cannon, Ed Chien, Sebastian Claici, Moon Duchin, Charlie Frogner, Jordan Ellenberg, Heng Guo, David Hayden, Pálvölgyi Dömötör Honlapja, Mamadou Moustapha Kanté, Fredrik Berg Kjolstad, Tianyu Liu, Aleksander Mądry, Elchanan Mossel, Marshall Mueller, David Palmer, Wes Pegden, Sebastien Roch, Mikhail Rudoy, Zach Schutzmann, Allan Sly, and Nike Sun.

We want to thank Jin-Yi Cai and Tianyu Liu for several in depth discussions that helped to guide this investigation, for catching some mistakes in an earlier version of §2.4 and for drawing our attention to relevant literature. We want to thank Sebastien Roch for several in depth discussions about bottlenecks in the flip walk, and asking useful questions which helped to guide the direction of investigation.

We also want to thank the creators, moderators and contributors to the Stack Exchange network; this project benefited greatly from the expertise of the individuals with whom that webpage connected us to: Heng Guo, Pálvölgyi Dömötör Honlapja, Mamadou Moustapha Kanté, Mikhail Rudoy, "Gamow," and "Kostya_I."

We also wish to thank the students from VRDI 2018. We especially thank the students in the graph partitioning group from week two, who suffered with us trying to find an algorithm that would solve the connected 2-partition uniform sampling problem: Austin Eide, Victor Eduardo Chavez-Heredia, Patrick Girardet, Amara Jaeger, Ben Klingensmith, Bryce McLaughlin, Heather Newman, Sloan Nietert, Anna Schall, Lily Wang. We also want to thank the team that developed Gerrychain at VRDI 2018, which we used extensively in §4: Mary Barker, Daryl DeFord, Robert Dougherty-Bliss, Max Hully, Anthony Pizzimenti, Preston Ward.

Funding. The first author was partially supported by the NSF RTG award DMS-1502553 and by U.S. National Science Foundation grants DMS-1107452, DMS-1107263, DMS-1107367. The authors acknowledge the generous support of NSF grant IIS-1838071 and the Prof. Amar G. Bose Research Grant. This work was partially completed at the Voting Rights Data Institute in the summer of 2018.

REFERENCES

[1] *Replication code*, https://github.com/LorenzoNajt/Code-For-Complexity-and-Geometry-of-Sampling-Connected-Graph-Partitions.
[2] *Stack exchange answer*, https://cstheory.stackexchange.com/a/41367/44995.
[3] *Stack exchange answer*, https://cstheory.stackexchange.com/a/42567/44995.
[4] *Stack exchange answer*, https://mathoverflow.net/a/313003/41873.
[5] *Stack exchange answer*, https://cstheory.stackexchange.com/a/43865/44995.
[6] *Stack exchange answer and discussion*, https://cstheory.stackexchange.com/a/41272/44995.
[7] *Stack exchange comment*, https://cstheory.stackexchange.com/q/41998/44995.
[8] *Stack exchange comments*, https://mathoverflow.net/q/316132/41873.
[9] *Stack exchange question*, https://cstheory.stackexchange.com/q/44338/44995.

[10] S. Aaronson, P $\stackrel{?}{=}$ NP, in Open problems in mathematics, Springer, 2016, pp. 1–122.
[11] H. Abbott and D. Hanson, *A lattice path problem*, Ars Combinatoria, 6 (1978), pp. 163–178.
[12] H. A. Akitaya, M. D. Jones, M. Korman, C. Meierfrankenfeld, M. J. Munje, D. L. Souvaine, M. Thramann, and C. D. Tóth, *Reconfiguration of connected graph partitions*, arXiv preprint arXiv:1902.10765, (2019).
[13] M. Altman, *Is automation the answer: The computational complexity of automated redistricting*, Rutgers Computer and Law Technology Journal, 23 (1997).
[14] S. Arnborg, J. Lagergren, and D. Seese, *Easy problems for tree-decomposable graphs*, Journal of Algorithms, 12 (1991), pp. 308–340.
[15] S. Arora and B. Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.
[16] S. Bangia, C. V. Graves, G. Herschlag, H. S. Kang, J. Luo, J. C. Mattingly, and R. Ravier, *Redistricting: Drawing the Line*, arXiv:1704.03360 [stat], (2017), http://arxiv.org/abs/1704.03360.
[17] A. Bar-Natan, L. Najt, and Z. Schutzman, *The gerrymandering jumble: Map projections permute districts' compactness scores*, arXiv preprint arXiv:1905.03173, (2019).
[18] R. Barnes and J. Solomon, *Gerrymandering and compactness: Implementation flexibility and abuse*, arXiv preprint arXiv:1803.02857, (2018).
[19] D. W. Barnette, *On steinitz's theorem concerning convex 3-polytopes and on some properties of planar graphs*, in The many facets of graph theory, Springer, 1969, pp. 27–40.
[20] P. Bergé, B. Mouscadet, A. Rimmel, and J. Tomasik, *Fixed-parameter tractability of counting small minimum (s,t)-cuts*, arXiv preprint arXiv:1907.02353, (2019).
[21] I. Bezáková, E. W. Chambers, and K. Fox, *Integrating and sampling cuts in bounded treewidth graphs*, in Advances in the Mathematical Sciences, Springer, 2016, pp. 401–415.
[22] H. L. Bodlaender and B. De Fluiter, *Parallel algorithms for series parallel graphs*, in European Symposium on Algorithms, Springer, 1996, pp. 277–289.
[23] V. Bouchitté, F. Mazoit, and I. Todinca, *Treewidth of planar graphs: connections with duality*, in Euroconference on Combinatorics, Graph Theory and Applications, vol. 10, 2001, pp. 34–38.
[24] M. Bousquet-Mélou, A. J. Guttmann, and I. Jensen, *Self-avoiding walks crossing a square*, Journal of Physics A: Mathematical and General, 38 (2005), p. 9159.
[25] U. C. Bureau, *Tiger/line shapefiles*, 2010, https://www2.census.gov/geo/tiger/TIGER2010/.
[26] S. Caldera, D. DeFord, M. Duchin, S. C. Gutekunst, and C. Nix, *Mathematics of nested districts: The case of alaska*, (2019), https://mggg.org/uploads/Alaska.pdf.
[27] E. W. Chambers, K. Fox, and A. Nayyeri, *Counting and sampling minimum cuts in genus g graphs*, Discrete & Computational Geometry, 52 (2014), pp. 450–475.
[28] G.-U. E. Charles et al., *Amicus brief of mathematicians, law professors, and students in support of the appellees and affirmance*, https://mggg.org/SCOTUS-MathBrief.pdf.
[29] J. Chen, *Expert report of Jowei Chen, ph.d.*, Raleigh Wake Citizen's Association et al. vs. The Wake County Board of Elections, (2017), https://www.pubintlaw.org/wp-content/uploads/2017/06/Expert-Report-Jowei-Chen.pdf.
[30] J. Chen and J. Rodden, *Unintentional Gerrymandering: Political Geography and Electoral Bias in Legislatures*, Quarterly Journal of Political Science, 8 (2013), pp. 239–269, https://www.nowpublishers.com/article/Details/QJPS-12033.
[31] M. Chikina, A. Frieze, and W. Pegden, *Assessing significance in a Markov chain without mixing*, Proceedings of the National Academy of Sciences, 114 (2017), pp. 2860–2864, http://www.pnas.org/content/114/11/2860.
[32] W. Cho and Y. Liu, *Toward a Talismanic Redistricting Tool: A Computational Method for Identifying Extreme Redistricting Plans*, Election Law Journal: Rules, Politics, and Policy, 15 (2016).
[33] W. K. T. Cho and Y. Y. Liu, *Sampling from complicated and unknown distributions: Monte Carlo and Markov Chain Monte Carlo methods for redistricting*, Physica A: Statistical Mechanics and its Applications, 506 (2018), pp. 170–178, http://www.sciencedirect.com/science/article/pii/S0378437118304163.
[34] B. Courcelle and J. Engelfriet, *Graph structure and monadic second-order logic: a language-theoretic approach*, vol. 138, Cambridge University Press, 2012.
[35] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized algorithms*, vol. 4, Springer, 2015.
[36] D. DeFord and M. Duchin, *Redistricting reform in virginia: Districting criteria in context*, Virginia Policy Review, 12(2) (2019), pp. 120–146.
[37] D. DeFord, H. Lavenant, Z. Schutzman, and J. Solomon, *Total Variation Isoperimetric Profiles*, arXiv:1809.07943 [cs, math], (2018), http://arxiv.org/abs/1809.07943.
[38] E. D. Demaine and M. Hajiaghayi, *The bidimensionality theory and its algorithmic applications*, The Computer Journal, 51 (2008), pp. 292–302.
[39] E. D. Demaine, M. Hajiaghayi, and D. M. Thilikos, *The bidimensional theory of bounded-genus graphs*, in International Symposium on Mathematical Foundations of Computer Science, Springer, 2004, pp. 191–203.
[40] T. S. C. O. P. M. District, *League of women voters of pennsylvania v. the commonwealth of pennsylvania*, https://www.brennancenter.org/sites/default/files/legal-work/LWV_v_PA_Majority-Opinion.pdf.
[41] M. Duchin, D. DeFord, and J. Solomon, *Recombination: A family of markov chains for redistricting (forthcoming)*.
[42] M. Duchin and B. E. Tenner, *Discrete geometry for electoral geography*, arXiv preprint arXiv:1808.05860, (2018).
[43] H. Duminil-Copin, G. Kozma, and A. Yadin, *Supercritical self-avoiding walks are space-filling*, in Annales de l'IHP Probabilités et statistiques, vol. 50, 2014, pp. 315–326.
[44] H. Duminil-Copin and S. Smirnov, *The connective constant of the honeycomb lattice equals $\sqrt{2+\sqrt{2}}$*, Annals of Mathematics, 175 (2012), pp. 1653–1665.

[45] M. Dyer and A. Frieze, *On the complexity of partitioning graphs into connected subgraphs*, Discrete Applied Mathematics, 10 (1985), pp. 139–153, http://linkinghub.elsevier.com/retrieve/pii/0166218X85900083.

[46] H.-D. Ebbinghaus, J. Flum, and W. Thomas, *Mathematical logic*, Springer Science & Business Media, 2013.

[47] J. Erickson, *Planar graphs*, http://jeffe.cs.illinois.edu/teaching/comptop/chapters/02-planar-graphs.pdf.

[48] M. Frick and M. Grohe, *The complexity of first-order and monadic second-order logic revisited*, Annals of pure and applied logic, 130 (2004), pp. 3–31.

[49] M. Garey, D. Johnson, and R. Tarjan, *The Planar Hamiltonian Circuit Problem is NP-Complete*, SIAM Journal on Computing, 5 (1976), pp. 704–714, https://epubs.siam.org/doi/10.1137/0205049.

[50] A. Gelman and C. Hennig, *Beyond subjective and objective in statistics*, Journal of the Royal Statistical Society: Series A (Statistics in Society), 180 (2017), pp. 967–1033.

[51] E. Grinbergs, *On planar regular graphs degree three without hamiltonian cycles*, (2009), https://arxiv.org/abs/arXiv:0908.2563.

[52] A. Grosse, J. Rothe, and G. Wechsung, *Relating partial and complete solutions and the complexity of computing smallest solutions*, in Italian Conference on Theoretical Computer Science, Springer, 2001, pp. 339–356.

[53] E. Győri, *On division of graphs to connected subgraphs*, North-Holland Publ. Comp, Amsterdam ; Oxford ; New York, 1978, pp. 485 – 494.

[54] G. Herschlag, H. S. Kang, J. Luo, C. V. Graves, S. Bangia, R. Ravier, and J. C. Mattingly, *Quantifying Gerrymandering in North Carolina*, (2018), http://arxiv.org/abs/1801.03783.

[55] G. Herschlag, R. Ravier, and J. C. Mattingly, *Evaluating Partisan Gerrymandering in Wisconsin*, arXiv:1709.01596 [physics, stat], (2017), http://arxiv.org/abs/1709.01596.

[56] T. R. Hoens, *Counting and sampling paths in graphs*, (2008).

[57] T. R. Hunter, *The first gerrymander? Patrick Henry, James Madison, James Monroe, and Virginia's 1788 congressional districting*, Early American Studies, (2011), pp. 781–820.

[58] R. Impagliazzo and A. Wigderson, P = BPP *unless E has subexponential circuits: derandomizing the XOR lemma*, in Proceedings of the 29th STOC, 1997, pp. 220–229.

[59] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, *Hamilton paths in grid graphs*, SIAM Journal on Computing, 11 (1982), pp. 676–686.

[60] T. Ito, X. Zhou, and T. Nishizeki, *Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size*, Journal of discrete algorithms, 4 (2006), pp. 142–154.

[61] I. Jensen, *A parallel algorithm for the enumeration of self-avoiding polygons on the square lattice*, Journal of Physics A: Mathematical and General, 36 (2003), p. 5731.

[62] I. Jensen, *Improved lower bounds on the connective constants for two-dimensional self-avoiding walks*, Journal of Physics A: Mathematical and General, 37 (2004), p. 11521.

[63] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani, *Random generation of combinatorial structures from a uniform distribution*, Theoretical Computer Science, 43 (1986), pp. 169–188, http://www.sciencedirect.com/science/article/pii/030439758690174X.

[64] T. Kennedy, *Monte carlo tests of stochastic loewner evolution predictions for the 2d self-avoiding walk*, Physical review letters, 88 (2002), p. 130601.

[65] R. Kenyon, *The asymptotic determinant of the discrete laplacian*, Acta Mathematica, 185 (2000), pp. 239–286.

[66] S. Khuller and V. V. Vazirani, *Planar graph coloring is not self-reducible, assuming* $P \neq NP$, Theoretical Computer Science, 88 (1991), pp. 183–189.

[67] R. Kueng, D. G. Mixon, and S. Villar, *Fair redistricting is hard*, Theoretical Computer Science, (2019).

[68] D. Lapoire, *Treewidth and duality for planar hypergraphs.*, (1996).

[69] G. F. Lawler, O. Schramm, and W. Werner, *On the scaling limit of planar self-avoiding walk*, arXiv preprint math/0204277, (2002).

[70] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*, American Mathematical Soc., 2009.

[71] Y. Y. Liu, W. K. T. Cho, and S. Wang, *PEAR: a massively parallel evolutionary computation approach for political redistricting optimization and analysis*, Swarm and Evolutionary Computation, 30 (2016), pp. 78–92, http://www.sciencedirect.com/science/article/pii/S2210650216300220.

[72] L. Lovasz, *A homology theory for spanning tress of a graph*, Acta Mathematica Hungarica, 30 (1977), pp. 241–251.

[73] N. Madras, *Critical behaviour of self-avoiding walks: that cross a square*, Journal of Physics A: Mathematical and General, 28 (1995), p. 1535.

[74] N. Madras and G. Slade, *The self-avoiding walk*, Springer Science & Business Media, 1996.

[75] D. B. Magleby and D. B. Mosesson, *A New Approach for Developing Neutral Redistricting Plans*, Political Analysis, 26 (2018), pp. 147–167.

[76] K. C. Martis, *The original gerrymander*, Political Geography, 8 (2008), pp. 833–839.

[77] J. Mattingly, *Declaration of Jonathan Mattingly*, Common Cause vs. Rucho, (2017), http://s10294.pcdn.co/wp-content/uploads/2016/05/Expert-Report-of-Jonathan-Mattingly.pdf.

[78] S. Montanari and P. Penna, *On sampling simple paths in planar graphs according to their lengths*, in International Symposium on Mathematical Foundations of Computer Science, Springer, 2015, pp. 493–504.

[79] T. M. D. of North Carolina, *Common cause v. rucho*, https://www.brennancenter.org/sites/default/files/legal-work/2018-08-27-142-Memorandum%20Opinion.pdf.

[80] W. Pegden, *Pennsylvania's congressional districting is an outlier: Expert report*, League of Women Voters vs. Pennsylvania General Assembly, (2017), https://www.brennancenter.org/sites/default/files/legal-work/LWV_v_PA_Expert_Report_WesleyPegden_11.17.17.pdf.

[81] A. Pönitz and P. Tittmann, *Improved upper bounds for self-avoiding walks in zd*, Electron. J. Combin, 7 (2000).

[82] D. Randall and A. Sinclair, *Self-testing algorithms for self-avoiding walks*, Journal of Mathematical Physics, 41 (2000), pp. 1570–1584.
[83] J. M. Schmidt, *Structure and constructions of 3-connected graphs*, PhD thesis, 2011.
[84] V. Shoup, *A computational introduction to number theory and algebra*, Cambridge university press, 2009.
[85] A. J. Sinclair, *Randomised algorithms for counting and generating combinatorial structures*, (1988).
[86] A. D. Sokal, *How to beat critical slowing-down: 1990 update*, Nuclear Physics B-Proceedings Supplements, 20 (1991), pp. 55–67.
[87] A. D. Sokal, *Monte carlo methods for the self-avoiding walk*, arXiv preprint hep-lat/9405016, (1994).
[88] A. D. Sokal and L. E. Thomas, *Absence of mass gap for a class of stochastic contour models*, Journal of Statistical Physics, 51 (1988), pp. 907–947.
[89] A. D. Sokal and L. E. Thomas, *Exponential convergence to equilibrium for a class of random-walk models*, Journal of Statistical Physics, 54 (1989), pp. 797–828.
[90] H. Suzuki, N. Takahashi, and T. Nishizeki, *A linear algorithm for bipartition of biconnected graphs*, Information Processing Letters, 33 (1990), pp. 227–231.
[91] I. S. Vicente and L. Najt, *Practical algorithms for counting and sampling simple cycles for graphs of bounded tree-width*, (Forthcoming).
[92] J. A. Wald and C. J. Colbourn, *Steiner trees, partial 2-trees, and minimum ifi networks*, Networks, 13 (1983), pp. 159–167.
[93] A. Wigderson, *The Complexity of the Hamiltonian Circuit Problem for Maximal Planar Graphs*, 1982, https://www.math.ias.edu/avi/node/820.
[94] D. B. Wilson, *Generating random spanning trees more quickly than the cover time*, in STOC, vol. 96, Citeseer, 1996, pp. 296–303.
[95] U. S. D. C. F. T. W. D. O. Wisconsin, *Whitford v. gill*, https://www.brennancenter.org/sites/default/files/legal-work/Whitford-Opinion112116.pdf.

Appendix A. Appendix for complexity results.

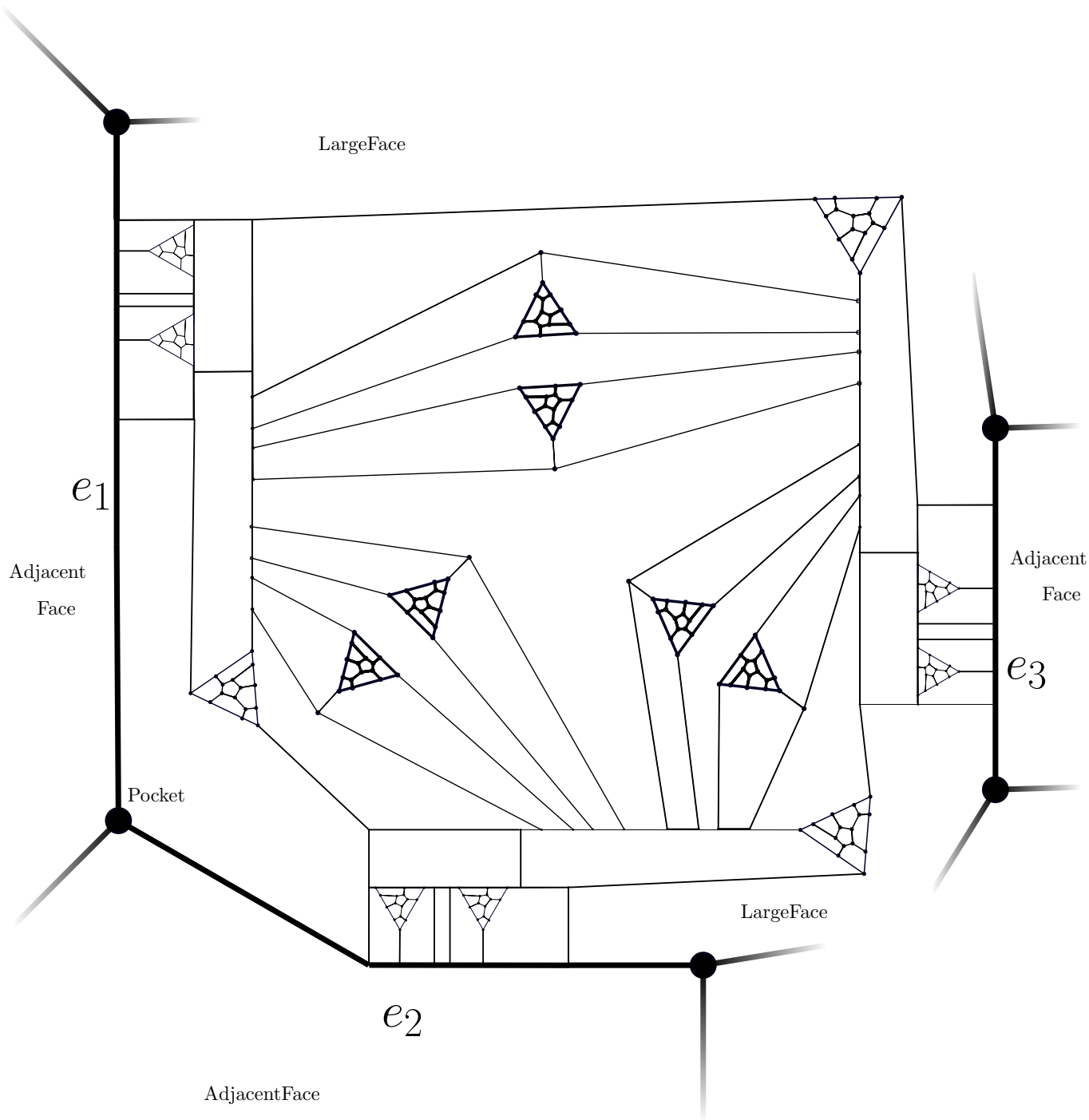


Figure A.1: The $3OR$ subdivision as it appears in Lemma 2.1.

A.1. Verifying Lemma 2.1.

A.2. Proving that R_d preserves $3CCP$ graphs. By construction, $R_d(G)$ (Definition 2.38) remains cubic, and $R_d(G)$ is planar if G is planar. The next few lemmas show that if G is 3-connected, so is $R_d(G)$.

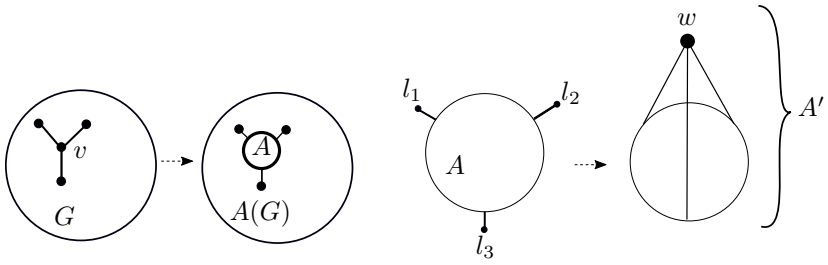


Figure A.2: Constructions described in Lemma A.1

We will let $\tilde{R}_d$ be the graph obtained from R_d by adding 3 leaf edges to each of $\{a_0, b_0, c_0\}$. The following lemma will show us that we can replace cubic vertices of G with copies of R_d and preserve 3-connectedness:

Lemma A.1. *Suppose that A is a graph with 3 leaf nodes, denoted by $L = \{l_1, l_2, l_3\}$. Let A' be the graph obtained by identifying the 3 leaf nodes of A. Let G be some graph with a cubic vertex $v \in V(G)$. Let $A(G)$ be a graph obtained from G by replacing v by A: that is, by deleting v from G and choosing some identification between the 3 leaf nodes of A and the 3 neighbors of v. Then, if G is 3 connected and A' is 3 connected, $A(G)$ is 3 connected.*

Proof. Suppose that $a, b, x, y \in V(A(G))$. We will show that there is a path in $A(G) \setminus \{a, b\}$ between x and y. Let $B = V(A) \setminus L$, and let $C : V(A(G)) \to V(G)$ be the map that contracts B back to v: for $s \notin B$, $C(s) = s$, and for $s \in B$, $C(s) = v$. There are two cases to consider:

1. If $C(x) \neq v$ and $C(y) \neq v$, then there is a path between $C(x)$ and $C(y)$ in $G \setminus \{C(a), C(b)\}$, since G is 3-connected. This can be lifted to a path between x and y in $A(G) \setminus \{a, b\}$.
2. If $C(x) = v$, it is always possible to find a path in $A(G) \setminus \{a, b\}$ from x to some x' with $C(x') \neq v$. There are three cases:
   (a) If $|L \cap \{a, b\}| = 0$: As A' is 3-connected, there is a path in $A' \setminus (B \cap \{a, b\})$ from x to w. This gives a path in $A(G)$ from x to a node L.
   (b) If $|L \cap \{a, b\}| = 1$: At most one node of $\{a, b\}$ can be contained in $B = A \setminus L$. Thus, $A' \setminus (B \cap \{a, b\})$ is 2-connected, so there are two paths in $A' \setminus (B \cap \{a, b\})$ from x to w. These give paths in $A(G)$, which only intersect $\{a, b\}$ at L, and of these paths connects to a node in $L \setminus \{a, b\}$.
   (c) If $|L \cap \{a, b\}| = 2$: Since A' is 3 connected, there are three node disjoint paths from x to w. In $A(G)$, this corresponds to a path to each of the three leaf nodes, one of which is not contained in $\{a, b\}$.

   Likewise, if $C(y) = v$, then we can connect y to some y', with $C(y') \neq v$. Once we have connected x to x' and y to y' outside of $A(v)$, we are back in Case 1.

The following lemma is well known; it is one of the Barnette-Grunbaum (BG) operations, introduced in [19, Proof of Theorem 2]. See also [83].

Lemma A.2 (BG-operation). *Let G be a 3-connected graph. Let e_1 and $e_2 \in E(G)$. Suppose that G' is the graph obtained from G by subdividing each e_i by introducing a vertex x_i, and then adding an edge from x_1 to x_2. Then G' is 3-connected.*

Lemma A.3. *If G is 3-connected, then so is $R_d(G)$.*

Proof. If we show that $(\tilde{R}_d)'$ (in the notation of Lemma A.1 and the paragraph preceding it) is 3 connected, then the claim follows from Lemma A.1 by considering $R_d(G)$ as obtained by replacing each node of G by an $\tilde{R}_d$ one at a time in the sense given by Lemma A.1. To prove that $(\tilde{R}_d)'$ is 3-connected we argue by induction. In the base case, $(\tilde{R}_0)'$ is a K_4 graph, so it is 3-connected. Let Q_d be obtained from $(\tilde{R}_d)'$ by adding a single node c in the center connected with three edges subdividing edges of the inner circle of R_d. If $(\tilde{R}_d)'$ is 3-connected, then it follows from applying the BG-operation of Lemma A.2 twice that Q_d is also 3-connected. From this it follows that $(\tilde{R}_{d+1})'$ 3-connected, because $(\tilde{R}_{d+1})'$ is obtained by replacing c with an $\tilde{R}_0$, so it is also 3-connected by Lemma A.1.

REMARK A.4. *Lemma A.1 and Lemma A.2 make it relatively straightforward to check that inserting the 3OR gadget preserves 3-connectedness, which is stated in [49] without proof.*

LEMMA A.5. *Suppose that H is a cubic planar graph, with face degree bounded by d. Then $R_d(H)$ has face degree bounded by $3d$.*

Proof. For each face, each vertex along that face supplies two additional edges when we replace vertices with copies of R_d. Thus, the face degree multiplies by 3. The claims follows. ∎

Altogether, we have shown that for all d, the construction $G \to R_d(G)$ sends $\mathscr{C}_m$ into $\mathscr{C}_{3m}$, where $\mathscr{C}_m$ is as in Definition 2.23.

A.3. Duality for connected k-partitions. In this section, we prove Theorem 2.52. There are three main steps in the proof, which answer three questions:

1. Can you recover a connected partition from its edge boundary?
2. What does the edge boundary of a partition of a plane graph look like in the dual graph?
3. In what way is the number of blocks of a connected partition reflected in its representation in the dual graph?

We state and prove the theorems that answer these questions in the next three subsections, and the end result is Theorem 2.52. The reader may note that 1) is answered by matroid duality between the graphic and cographic matroids (specifically between flats, which correspond to connected partitions, and unions of circuits)[17], that the answer to 2) follows quickly from the usual bond-cycle duality, and that 3) is a discrete version of Alexander duality.

A.3.1. Connected partitions and edge cuts. We first recall the bijection between connected partitions and edge-cuts:

DEFINITION A.6 (Unordered connected partitions). *Let $\mathscr{P}(G)$ denote the set of* unordered *partitions of $V(G)$. Let $\mathscr{P}^c(G) \subseteq \mathscr{P}(G)$ denote the set of partitions such that each block induces a connected subgraph. That is, $\mathscr{P}^c(G) = \bigcup_{k=1}^{|V(G)|} \mathscr{P}_k(G)$.*

DEFINITION A.7 (Edge cut). *Let P be a partition of $V(G)$. If $P = \{A_1, \dots, A_k\}$, then we refer to the A_i as the* blocks *of P. Let $\mathrm{cut}(P)$ denote the set of edges of G with endpoints in different blocks of P.*

DEFINITION A.8 (Cut sets). *Let $\mathrm{Cuts}(G)$ be the set of the cuts of partitions of $V(G)$. That is, $\mathrm{Cuts}(G) = \{\mathrm{cut}(P) : P \in \mathscr{P}(G)\}$. The elements of $\mathrm{Cuts}(G)$ are called cut sets.*

DEFINITION A.9 (Component map). *Given $J \in \mathrm{Cuts}(G)$, define a partition $\mathrm{comp}(J) \in \mathscr{P}^c(G)$ as the connected components of $G \setminus J$. This defines a function $\mathrm{comp} : \mathrm{Cuts}(G) \to \mathscr{P}^c(G)$.*

PROPOSITION A.10. *The functions* comp *and* cut *induce a bijection between $\mathrm{Cuts}(G)$ and $\mathscr{P}^c(G)$.*

Proof. To show that cut is surjective, we observe that if P is any partition, we can define a connected partition P', whose blocks are the connected components of the blocks of P, and $\mathrm{cut}P = \mathrm{cut}P'$. We will conclude by showing that $\mathrm{comp} \circ \mathrm{cut} = id$. First, observe that $\mathrm{comp} \circ \mathrm{cut}$ does not merge any blocks, since every path in G between two blocks has to cross a cut edge. Second, observe that $\mathrm{comp} \circ \mathrm{cut}$ does not split any blocks, since two points in any block *of a connected partition* are always connected by a path that does not use any cut edges.

So far we have established that a connected partition is determined by the boundaries between its blocks. Next, we work towards characterizing the shapes that can arise as such boundaries, by treating them as subgraphs of the planar dual.

A.3.2. Dual connected partitions and connected partitions. The following straightforward lemma is useful for proving the duality theorem:

LEMMA A.1. *Let G be a graph, and $J \subseteq E(G)$. Then each connected component of $G[J]$ is two edge-connected if and only if each connected component of $G[J]$ has no bridge edges if and only if $G[J]$ is a union of not-necessarily disjoint simple cycles.*

[17] We wish to acknowledge a helpful MathOverflow discussion that drew our attention to this connection to matroids [8].

Definition A.11 (Dual connected partitions). *Let $\mathscr{E}_2(G)$ denote the set of subsets of edges of G that are unions of not-necessarily disjoint simple cycles. We will call these the* dual connected partitions.

The purpose of the next few propositions is to show that dual connected partitions are plane duals of the cuts of connected partitions. First we recall the bijection between the edges of a plane graph and the edges of its dual:

Definition A.12 (Dual edges). *Let G be a plane graph. For an edge $e \in E(G)$, let e^* denote the edge in G^* with the property that the two endpoints of e^* are the shores of e, i.e., the two faces that are separated by e. For a set $J \subseteq E(G)$, denote by J^* the corresponding set of edges in G^*. We define a function $D(J) = J^*$, which is a bijection $2^{E(G)} \to 2^{E(G^*)}$*

We aim to prove a plane duality between $\text{Cuts}(G)$ and $\mathscr{E}_2(G^*)$. In particular, we want show that D induces a bijection between $\text{Cuts}(G)$ and $\mathscr{E}_2(G^*)$. Towards that, we will recall the plane duality between even subgraphs and the edge boundaries, which will be useful for controlling the topology of $D(J)$ for $J \in \text{Cuts}(G)$.

Definition A.13 (Edge boundary). *Let $G = (V, E)$ be a graph, and $A \subseteq V$. Denote by $\text{cut}(A) = \text{cut}(\{A, A^c\}) = \partial_E(G)$, the edge boundary of A.*

Definition A.14 (Even Subgraphs). *Let $G = (V, E)$ be a graph. A subset $J \subseteq E$ defines an even subgraph $G[J]$ if the degree of each node of $G[J]$ is even. Let $Even(G) = \{J \subseteq 2^{E(G)} : G[J] \text{ is an even subgraph}\}$.*

Proposition A.15 (Proposition 2.1 in [47][18]). *Let G be a connected plane graph, and let $H \subseteq E(G)$. Then, H is an even subgraph if and only if H^* is an edge boundary. Moreover, H is a simple cycle if and only if H^* is the cut of a connected 2-partition.*

The following well-known lemma will be useful for relating $Even(G)$ to $\mathscr{E}_2(G)$:

Lemma A.16 (Euler). *Let G be a graph. Then $J \subseteq E(G)$ is an even subgraph if and only if J is a union of pairwise disjoint simple cycles.*

The previous theorem characterized edge boundaries, which are the cut sets of not-necessarily connected 2-partitions, using the planar dual. The next proposition will characterize the cut sets of connected k-partitions using the planar dual.

Proposition A.17. *Let G be a connected plane graph. Let $H \subset E(G)$. Then $H \in \mathscr{E}_2(G)$ if and only if H^* is a cut set. In particular, D gives a bijection between $\mathscr{E}_2(G)$ and $\text{Cuts}(G^*)$.*

Proof. $\Rightarrow$ Suppose that $H \in \mathscr{E}_2(G)$, and let $P \in \mathscr{P}^c(G^*)$ be the connected partition defined by the connected components of $G^* \setminus H^*$. We show that $\text{cut}(P) = H^*$. Let $e \in H$, and let $C \subseteq H$ be a cycle containing it. Then the shores of e are necessarily in different components of $G^* \setminus C^*$ by Proposition A.15, and thus in different components of $G^* \setminus H^*$. Thus, $e^* \in \text{cut}(P)$, so $H^* \subseteq \text{cut}(P)$. On the other hand, if $e^* \in \text{cut}(P)$, then the shores of e are in different components of $G^* \setminus H^*$, so every path in G^* between the two shores of e must pass through H^*. In particular, the path e^* must pass through H^*, which means that $e^* \in H^*$. Thus, $\text{cut}(P) \subseteq H^*$.

$\Leftarrow$ Now suppose that $H^* = \text{cut}(P)$ for some $P \in \mathscr{P}^c(G)$. Take any $e \in H$. We want to show that e is not a bridge edge. Suppose that A and B are the blocks of P containing the faces in the two shores of e. Now, create a not necessarily connected 2-partition P' by reassigning all of the blocks of P that are not A or B to be part of block A. Now, $\text{cut}(P') \subseteq \text{cut}(P)$ and $e \in \text{cut}(P')$. Since $\text{cut}(P')^*$ is a union of edge disjoint cycles (Lemma A.16), there is a simple cycle C in $\text{cut}(P')^*$ that contains e. In particular, $\text{cut}(P)^* = H \supseteq C$, so e could not have been a bridge edge of H. ∎

We summarize the previous two results in a single duality statement:

Proposition A.18 (Duality between connected partitions and dual connected partitions). *The functions $\text{comp} \circ D^{-1}$ and $D \circ \text{cut}$ are mutual inverses, inducing a bijection between $\mathscr{P}^c(G)$ and $\mathscr{E}_2(G^*)$.*

Proof. Since D induces a bijection between $\mathscr{E}_2(G^*)$ and $\text{Cuts}(G)$ (Proposition A.17) and comp and cut give a bijection between $Cuts(G)$ and $\mathscr{P}^c(G)$ (Proposition A.10), the claim follows. ∎

[18] Beware that his terminology is different from ours; specifically, he refers to what we call an edge boundary as an edge cut.

A.3.3. The number of blocks and the circuit rank. Now we will review some facts that relate the number of blocks in a connected partition of a plane graph to the circuit rank of the corresponding dual connected partition.

DEFINITION A.19 (h_1 and h_0). *Let G be a graph. Then $h_1(G)$ denotes the circuit rank of G, $h_0(G)$ denotes the number of connected components of G.*

PROPOSITION A.20. *Let $G = (V, E)$ be a plane graph. Then $1 + h_0(G) = |V(G)| - |E(G)| + |F(G)|$.*

Proof. One can add $h_0 - 1$ edges connecting the components, without changing the number of faces. If the new graph has E' edges, then $E' = E + h_0 - 1$, and we have $V - E' + F = 2$, from which the formula follows.

PROPOSITION A.21. *If G is a graph, then $h_1(G) - h_0(G) = |E(G)| - |V(G)|$.*

Proof. Since h_1 is the cycle rank, which is the dimension of the kernel of the boundary map $\partial : \mathbb{F}_2^E \to \mathbb{F}_2^V$, and h_0 is the rank of the cokernel of ∂, this is just a statement of the rank-nullity theorem. ∎

PROPOSITION A.22. *Let G be a connected plane graph. For $P \in \mathscr{P}^c(G)$ and $J \in \mathscr{E}_2(G)$, $h_1(G^*[\mathrm{cut}(P)^*]) = |P| - 1$ and $|\mathrm{comp}(J^*)| - 1 = h_1(G[J])$. Here $|P|$ counts the number of blocks of P.*

Proof. Let $P \in \mathscr{P}^c(G)$. Let $J = \mathrm{cut}(P)^*$. Proposition A.20 and Proposition A.21 together yield that $h_1(G^*[J]) = h_0 - V + E = |F(G^*[J])| - 1$. Since the number of faces of $G^*[J]$ is the number of components of $G \setminus J$, and since $\mathrm{comp}(J) = \mathrm{comp} \circ \mathrm{cut}(P) = P$, we obtain $h_1(G^*[\mathrm{cut}(P)^*]) = |P| - 1$. Now consider $J \in \mathscr{E}_2(G^*)$, and let $P = \mathrm{comp}(J^*)$. Since $J = \mathrm{cut}(P)^*$, it follows from $h_1(G^*[\mathrm{cut}(P)^*]) = |P| - 1$ that $h_1(G^*[J]) = |P| - 1$, so the claim follows from $(G^*)^* = G$. ∎

Finally, we present the duality theorem:

DEFINITION A.23 (Dual k-partition). *We define $P_k^*(G) = \{J \in \mathscr{E}_2(G) : h_1(G[J]) = k - 1\}$. We call the elements of this set dual k-partitions.*

THEOREM A.24 (Duality between $P_k(G)$ and $P_k^*(G^*)$). *The map $D \circ \mathrm{cut} : P_k(G) \to P_k^*(G^*)$ is a bijection, with $\mathrm{comp} \circ D^{-1} : P_k^*(G^*) \to P_k(G)$ as its inverse. Both are computable in polynomial time.*

Proof. The bijection follows from Proposition A.22 and Proposition A.18. It is well known that D and comp and cut can be computed in polynomial time. ∎

Appendix B. Positive results.

B.1. Using marginal counts. In this section, we prove the correctness of Algorithm 5.1.

PROPOSITION B.1. *The output of Algorithm 5.1 is a random variable J valued in $2^{[n]}$ and drawn with distribution p. Moreover, if a call to the oracle O takes time $O(T(n))$, then the total runtime of Algorithm 5.1 is $O(nT(n))$.*

Proof. Let S be a random variable distributed according to p. Let $\mathbb{P}$ be the probability measure underlying the process of the algorithm and the random variable S. Let J_k be the random set J on the kth step of Algorithm 5.1. Using induction, we will show that, for all $m \in [n]$,

$$\mathbb{P}(J_m = W) = \mathbb{P}(S \cap [m] = W). \tag{B.1}$$

The desired conclusion is the case $m = n$. In the base case, when $k = 1$, Equation (B.1) holds because $\mathbb{P}(J_1 = \{1\}) = p(1|\emptyset) = \mathbb{P}(1 \in S) = \mathbb{P}(S \cap \{1\} = \{1\})$. Now, suppose that for some $m \geq 1$ with $m < n$, it holds that $\mathbb{P}(J_m = W) = \mathbb{P}(S \cap [m] = W)$ for all $W \subset [m]$. Recall that from the definition of Algorithm 5.1 we have that $\mathbb{P}(J_{m+1} = W \cup \{m+1\} | J_m = W) = \mathbb{P}(m + 1 \in S | S \cap [m] = W)$.

The inductive step now follows by a computation:

$$\begin{aligned}
\mathbb{P}(J_{m+1} = W \cup \{m+1\}) &= \mathbb{P}(J_{m+1} = W \cup \{m+1\} | J_m = W))\mathbb{P}(J_m = W) \\
&= \mathbb{P}(m+1 \in S | S \cap [m] = W)\mathbb{P}(S \cap [m] = W) \\
&= \mathbb{P}(S \cap [m+1] = W \cup \{m+1\}).
\end{aligned}$$

Likewise $\mathbb{P}(J_{m+1} = W) = \mathbb{P}(S \cap [m+1] = W)$. ∎

B.2. Series-parallel graphs. We recall the definition of a series-parallel graph, a class of graphs well suited to dynamic programming algorithms.

DEFINITION B.2 (Two-terminal graphs). *A two-terminal graph G is a graph with two distinguished nodes: a source, $\sigma(G)$, and a sink $\tau(G)$. A pair of two-terminal graphs, G and H, are said to be isomorphic, $G \cong H$, if there is an isomorphism of the underlying graphs which maps the source to the source and the sink to the sink.*

EXAMPLE B.3. *The complete graph on $\{0,1\}$ is naturally a two-terminal graph, where we set $\sigma(K_2) = 0$ and $\tau(K_2) = 1$. We denote it by K_2*

DEFINITION B.4 (Series Composition). *Let G_1 and G_2 be two-terminal graphs. We define $G_1 \circ G_2$ as the graph obtained from the disjoint union of G_1 and G_2 by identifying τ_1 and σ_2, and we make it into a two-terminal graph by setting $\sigma(G_1 \circ G_2) = \sigma(G_1)$ and $\tau(G_1 \circ G_2) = \tau(G_2)$.*

DEFINITION B.5 (Parallel Composition). *In the notation of Definition B.4, we define $G_1 \parallel G_2$ as the graph obtained from the disjoint union of G_1 and G_2 by making the identifications $\sigma_1 \sim \sigma_2$ and $\tau_1 \sim \tau_2$. We make $G_1 \parallel G_2$ into a two-terminal graph by defining $s(G_1 \parallel G_2) = [\sigma(G_1)] = [\sigma(G_2)]$ and $t(G_1 \parallel G_2) = [\tau(G_1)] = [\tau(G_2)]$.*

DEFINITION B.6 (Series-Parallel graphs). *We define the class of series-parallel graphs as the smallest class of two-terminal graphs that is closed under Parallel Composition and Series Composition, and which contains the two-terminal graph K_2.*

The feature which makes series-parallel graphs convenient for dynamic programming is that we can record the series and parallel composition operations into a tree, called the SP-tree, around which we can organize dynamic programs:

DEFINITION B.7 (SP-tree). *An SP tree is a rooted binary tree, where the children of any internal node have an ordering, and where each internal node of the tree is labelled P or S. We assign to each leaf a copy of the two-terminal graph K_2. Then, to each internal node we assign the graph obtained by applying either Series Composition or Parallel composition to its children, depending on the label of that internal node; for the series composition (labelled S), the order of the composition is in the order on the children prescribed by the tree, and for parallel composition (labelled P) the order does not matter. If T is an SP-tree, define $G(T)$ as the two-terminal graph assigned to the root of T. If $X = (G, \sigma, \tau)$ is a series-parallel graph, we say that an SP-tree T is an SP-tree for X if $X \cong G(T)$.*

LEMMA B.8 (Theorem 4.1 of [22]). *Given a graph G, determining if G is series-parallel and if so building an SP-tree for it can be done in linear time.*

B.3. Computing marginal probabilities on graphs of treewidth 2. In this section we give a polynomial time algorithm for counting simple cycles on graphs of treewidth 2. We also prove as a byproduct of the method that it is possible to efficiently sample from a much broader family of distributions than just uniform, namely those defined by Definition 5.8. For these computations, it will be convenient to extend the concept of a network to allow edge weights in other rings. These results are extended in [91] to graphs of bounded treewidth.

DEFINITION B.9 (R-network). *Let G be a graph and R a ring, and let $w : E(G) \to R$ be a function. Then we will call (G, w) an R-network.*

Let $\mathbb{Q}$ be the rationals. Let (G, w) be a $\mathbb{Q}$-network with non-negative weights, and let $J, J' \subseteq E(G)$. For sampling with Algorithm 5.1 we would like to be able to compute the total ν_w (Definition 5.8) mass of the simple cycles containing J and disjoint from J'. The approach here will be to encode that mass as the evaluation of a generating function (Definition B.10), which we can evaluate efficiently on series-parallel graphs by dynamic programming.

DEFINITION B.10 (Simple cycle generating function). *Let (G, w) be an R-network. Let $f_{SC}(G, w) := \sum_{C \in SC(G)} \prod_{e \in C} w(e)$ denote the generating function of the simple cycles of G evaluated at the weights w.*

Let (G, c) be a $\mathbb{Q}$-network. To sample from ν_c of G, the marginals we need in the course of Proposition B.1 are easily computed from $N_c(\{C \in SC(G) : C \cap J' = \emptyset, C \supseteq J\}) = \sum_{C \in SC(G), C \cap J' = \emptyset, C \supseteq J} \prod_{e \in C} c(e)$. To

obtain these measurements for any given J, J', let x be some formal variable, and set $w(e) = xc(e)$ for $e \in J$, $w(e') = 0$ for $e' \in J'$ and $w(f) = c(e)$ otherwise. Then the coefficient of the $x^{|J|}$ term of $f_{SC}(G, w)$ is $\sum_{C \in SC(G): J \subseteq C, J' \cap C = \emptyset} \prod_{e \in C} c(e)$, which is the N_c mass of all the simple cycles that are disjoint from J' and that contain J. We next show that we can compute $f_{SC}(G, w)$ *if* G is a series-parallel graph, via a dynamic programming algorithm which runs in time polynomial in $|G|$ and $|w|$. We also need to keep track of the corresponding generating function for the simple paths, which we define next.

DEFINITION B.11 (Simple path generating function). *Let (G, w) be a series-parallel R-network, with source σ and sink τ. Then we define $f_{SP}(G) = \sum_{\gamma \in SP_{\sigma,\tau}(G)} \prod_{e \in \gamma} w(e)$, where $SP_{\sigma,\tau}(G)$ is the set of simple paths from σ to τ in G, where a path is a* sequence *of* edges.

LEMMA B.12. *Let (G_1, w_1) and (G_2, w_2) be series-parallel R-networks. Let $w : E(G_1) \sqcup E(G_2) \to R$ be the unique weight function that restricts to w_1 and w_2. Then, let w make both $G_1 \circ G_2$ and $G_1 \parallel G_2$ into R-networks, using that both have edge set $E(G_1) \cup E(G_2)$. Then:*

$$f_{SC}(G_1 \circ G_2, w) = f_{SC}(G_1, w_1) + f_{SC}(G_2, w_2) \tag{B.2}$$

$$f_{SP}(G_1 \circ G_2, w) = f_{SP}(G_1, w_1) f_{SP}(G_2, w_2) \tag{B.3}$$

$$f_{SC}(G_1 \parallel G_2, w) = f_{SC}(G_1, w_1) + f_{SC}(G_2, w_2) + f_{SP}(G_1, w_1) f_{SP}(G_2, w_2) \tag{B.4}$$

$$f_{SP}(G_1 \parallel G_2, w) = f_{SP}(G_1, w_1) + f_{SP}(G_2, w_2) \tag{B.5}$$

Proof. In each case, the equality on generating functions will follow from a bijection of sets.

Proof of Equation (B.2). $SC(G_1 \circ G_2) = SC(G_1) \cup SC(G_2)$.

Proof of Equation (B.3). $SP(G_1 \circ G_2) = \{\gamma_1 \circ \gamma_2 : \gamma_i \in SP(G_i)\}$, where $\circ$ between paths denotes concatenation.

Proof of Equation (B.4). $SC(G_1 \parallel G_2) = SC(G_1) \cup SC(G_2) \cup \{Set(\gamma_1 \circ Reverse(\gamma_2)) : \gamma_i \in SP(G_i)\}$, where the *Reverse* of a path is that path run backwards, and $Set(\_)$ takes the sequence of edges and turns it into a subset of $E(G_1 \parallel G_2)$.

Proof of Equation (B.4). $SP(G_1 \parallel G_2) = SP(G_1) \cup SP(G_2)$

Now we recall a key lemma that will allow us to reduce the treewidth 2 case to the series-parallel case, and prove the theorem about efficiently evaluating f_{SC}:

LEMMA B.13 ([92] Theorem 4.1). *If H is a graph of treewidth ≤ 2, then there is a series-parallel graph G and an embedding $i : H \to G$. Both H and i are computable from H in linear time.*

LEMMA B.14. *Suppose (H, w) is a $\mathbb{Q}[x]$-network with treewidth ≤ 2. Then $f_{SC}(H, w)$ can be computed in time polynomial in $|(H, w)|$. In particular, if (H, c) is a $\mathbb{Q}$-network, then $N_c(\{C \in SC(H) : C \cap J' = \emptyset, C \supseteq J\})$ can be computed in time polynomial in $|(H, c)|$.*

Proof. By Lemma B.13, there is a series-parallel graph G, which contains H as a subgraph, and moreover G and $i : H \to G$ can be constructed in linear time. Extend w to all the edges of G by defining $w(e) = 0$ for $e \in E(G) \setminus E(H)$. Then we have that $f_{SC}(G, w) = f_{SC}(H, w)$ since all the cycles of G that are not cycles of H have weight zero. We let T_G be a binary SP-tree of G, which has $O(|G|)$ nodes. Using Lemma B.12, we compute f_{SC} and f_{SP} at each node. The cost of the calculation at each node is bounded by the cost of multiplying and adding the corresponding generating functions for the nodes children. Each generating function has degree at most $O(|G|(\max_{e \in E} \deg(w(e)))$, and has coefficients that have binary encoding whose length is polynomial in $|(G, w)|$. Thus, the first claim follows. The second claim follows because if we set $w(e) = c(e)x1_{x \in J} + c(e)1_{x \notin J' \cup J}$, then the coefficient of the $x^{|J|}$ term of $f_{SC}(G, w)$ is $N_c(\{C \in SC(H) : C \cap J' = \emptyset, C \supseteq J\})$. ∎

B.4. Dynamic program for counting balanced partitions on series-parallel graphs. In this section we show how to set up a dynamic program that will count the number of balanced connected 2-partitions of a series-parallel graph. We will be interested in the case where nodes have weights valued in $\mathbb{N} = \{0, 1, 2, \ldots\}$, but it will be convenient extend these weights to take values in a larger monoid, N, which will also keep track of when a set of nodes is non-empty.

DEFINITION B.15 (The monoid N). *Let E be the commutative monoid given by $E = \{\{\emptyset, \neg\emptyset\}, \cup, \emptyset\}$ where $\emptyset$ is the identity element, and $\neg\emptyset \cup \neg\emptyset = \neg\emptyset$. Let $\mathbb{N}$ be the monoid of natural numbers with addition. Let $N = \mathbb{N} \times E$ and by abuse of notation we let $0 \in N$ denote the additive identity, and $+$ the binary operation in N. Let $n : N \to \mathbb{N}$ and $e : N \to E$ be the natural projections.*

DEFINITION B.16 ((Admissible) node-weighted graphs). *(G, w) will denote a graph $G = (V, E)$ along with a function $w : V(G) \to N$. If $e(w(v)) = \neg\emptyset$ for all $v \in V$, then we call (G, w) admissible. For any $A \subseteq V(G)$, define $w(A) = \sum_{a \in A} w(a)$.*

DEFINITION B.17 (The DP-table $X(G, w)$). *For a series-parallel graph G with source σ and sink τ, and weight function $w : V(G) \to N$, we imitate [45] and define:*

$$X(G, w) = \{(a_1, a_2, a_3, m) \in N^3 \times \mathbb{N} | \textit{there are exactly } m \textit{ partitions of } V(G) \textit{ into blocks } V_1, V_2, V_3 \textit{ such that:}$$
$$\textit{(i) } w(V_i) = a_i \textit{ and } G[V_i] \textit{ is connected for } i = 1, 2, 3$$
$$\textit{(ii) } \sigma \in V_1$$
$$\textit{(iii) } a_3 = 0 \textit{ implies that } \tau(G) \in V_1 \textit{ and } a_3 \neq 0 \textit{ implies that } \tau(G) \in V_3. \ \}$$

That is, $X(G, w)$ is a function $N^3 \to \mathbb{N}$ that counts the number of partitions of G into connected 3-partitions which blocks of given weights. If we know $X(G, w)$, we will be able to calculate $|P_2^0(G, w)|$ (Theorem B.23).

DEFINITION B.18 (Series and parallel composition for weighted SP graphs). *Let (G_1, w_1) and (G_2, w_2) be series-parallel graphs with N valued weights on the nodes. Define weights w on the nodes of $G = G_1 \circ G_2$ and $G = G_1 \parallel G_2$ by first thinking of w_1 and w_2 as functions on $V(G)$, through extending them to the other nodes by assigning them the value $(0, \neg\emptyset)$, and then setting $w = w_1 + w_2$.*

DEFINITION B.19 (Naming conventions for series-parallel compositions). *In the case of $G_1 \circ G_2$, we let ϵ denote the node $\tau_1 = \sigma_2$, $\sigma = \sigma_1$ and $\tau = \tau_2$. In the case of $G_1 \parallel G_2$, $\sigma = \sigma_1 = \sigma_2$ and $\tau = \tau_1 = \tau_2$.*

Let (G, w) be a node-weighted series-parallel graph. The next several propositions will show that we can compute $X(G, w)$ by a dynamic programming algorithm on a binary SP-tree of G. Specifically, we will show how to compute $X((G_1, w_1) \circ (G_2, w_2))$ and $X((G_1, w_1) \parallel (G_2, w_2))$ from $X(G_1, w_1)$ and $X(G_2, w_2)$, using a algorithms that are polynomial time in G and pseudopolynomial time in the total weights. To prove these algorithms to be correct, we will compare the dynamic calculation of $X(G, w)$ with a dynamic enumeration of the partitions in question, using the following definition:

DEFINITION B.20 (The enumeration version of $X(G, w)$).

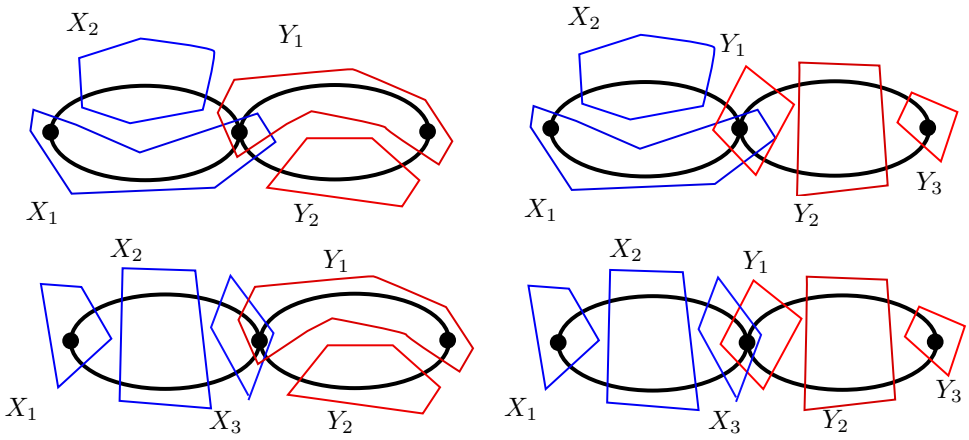
$$\widetilde{X}(G) = \{(V_1, V_2, V_3) \in \mathcal{P}(V(G))^3 : (V_1, V_2, V_3) \textit{ form a partition of } V(G) \textit{ such that}$$
$$\textit{(i) } G[V_i] \textit{ is connected for } i = 1, 2, 3$$
$$\textit{(ii) } \sigma \in V_1$$
$$\textit{(iii) } V_3 = \emptyset \textit{ implies that } \tau(G) \in V_1 \textit{ and } V_3 \neq \emptyset \textit{ implies that } \tau(G) \in V_3. \ \}$$

In the proof of correctness for the dynamic program for evaluating $X((G, w))$, we will explain how to compute $\tilde{X}(G_1 \circ G_2)$ from $\tilde{X}(G_1)$ and $\tilde{X}(G_2)$ by merging the blocks sharing the node ϵ, and accepting the output when it results in an element of $\tilde{X}(G_1 \circ G_2)$.

Algorithm B.1 `SeriesPartitions`

Input: $X((G_1, w_1))$ and $X((G_2, w_2))$ for (G_1, w_1) and (G_2, w_2).
Output: $X((G_1, w_1) \circ (G_2, w_2))$

Set f as constantly zero on $\{(a_1, a_2, a_3) \in \mathbb{N}^3 : 0 \leq a_i, \sum a_i = w((G_1, w_1) \circ (G_2, w_2))\}$.
for $(a_1, a_2, a_3, m_1) \in X(G_1, w_1)$ and $(b_1, b_2, b_3, m_2) \in X(G_2, w_2)$ **do**
  if ($a_3 = 0$ and $b_3 = 0$) and ($a_2 = 0$ or $b_2 = 0$) **then**
    $f(a_1 + b_1, a_2 + b_2, 0)$ += $m_1 m_2$
  if ($a_3 = 0$ and $b_3 \neq 0$) and ($a_2 = 0$ or $b_2 = 0$) **then**
    $f(a_1 + b_1, a_2 + b_2, b_3)$ += $m_1 m_2$
  if ($a_3 \neq 0$ and $b_3 = 0$) and ($a_2 = 0$ or $b_2 = 0$) **then**
    $f(a_1, a_2 + b_2, a_3 + b_1)$ += $m_1 m_2$
  if $a_3 \neq 0$ and $b_3 \neq 0$ and ($a_2 = 0$ and $b_2 = 0$) **then**
    $f(a_1, a_3 + b_1, b_3)$ += $m_1 m_2$
Return f



(a) Case 1 (b) Case 2

(c) Case 3 (d) Case 4

Figure B.1: The four cases in Algorithm B.1

Proposition B.21. *If (G, w) is admissible, then Algorithm B.1 runs correctly and in polynomial time in $(|G|, n(w(G)))$.*[19]

Proof. We need to check that each element of $P_2(G)$ is counted exactly once. To verify this, we explain another algorithm, Algorithm B.2, that computes $\widetilde{X}(G_1 \circ G_2)$ from $\widetilde{X}(G_1)$ and $\widetilde{X}(G_2)$, but in exponential time. We will verify that in the course of this algorithm each element of $\widetilde{X}(G_1 \circ G_2)$ is computed exactly once. Finally, we will explain that the correctness of Algorithm B.1 can be seen by coupling it with a accelerated version of Algorithm B.2, and we compute the time it takes for Algorithm B.1 to run.

[19] This is pseudopolynomial in the total weight.

Algorithm B.2 `SetLevelSeriesPartitions`

Input: series-parallel graphs G_1 and G_2 along with sets $\widetilde{X}(G_1)$ and $\widetilde{X}(G_2)$
Output: $\widetilde{X}(G_1 \circ G_2)$

Initialize F as the zero function on $\mathcal{P}(V(G))^3$, where $\mathcal{P}$ denotes the powerset.
for $(X_1, X_2, X_3) \in \widetilde{X}(G_1)$ and $(Y_1, Y_2, Y_3) \in \widetilde{X}(G_2)$ **do**
  if $X_3 = \emptyset$ AND $Y_3 = \emptyset$ AND $(X_2 = \emptyset$ OR $Y_2 = \emptyset)$ **then**
    $F(X_1 \cup Y_1, X_2 \cup Y_2, \emptyset)$ += 1
  if $X_3 = \emptyset$ AND $Y_3 \neq \emptyset$ AND $(X_2 = \emptyset$ OR $Y_2 = \emptyset)$ **then**
    $F(X_1 \cup Y_1, X_2 \cup Y_2, Y_3)$ += 1
  if $X_3 \neq \emptyset$ AND $Y_3 = \emptyset$ AND $(X_2 = \emptyset$ OR $Y_2 = \emptyset)$ **then**
    $F(X_1, X_2 \cup Y_2, X_3 \cup Y_1)$ += 1
  if $X_3 \neq \emptyset$ AND $Y_3 \neq \emptyset$ AND $X_2 = \emptyset$ AND $Y_2 = \emptyset$ **then**
    $F(X_1, X_3 \cup Y_1, Y_3)$ += 1
Return F

Our first goal is to verify that the function F returned by Algorithm B.2 is the indicator of $\tilde{X}(G)$ in $\mathcal{P}(V(G))^3$. First, it is straightforward to check that $\mathrm{supp}(F) \subseteq \tilde{X}(G)$, by examining each case. Let $(Z_1, Z_2, Z_3) \in \tilde{X}(G)$. There are four cases based on how Z separates (σ, ϵ, τ). The cases are:
1. There is a block containing σ, ϵ and τ.
2. There is a block containing σ and ϵ, and a distinct block containing τ.
3. There is a block containing σ, and a distinct block containing ϵ and τ.
4. σ, ϵ and τ are each in a different block.

This accounts for 4 of the 5 equivalence relations on $\{\sigma, \epsilon, \tau\}$. The missing equivalence relation on $\{\sigma, \epsilon, \tau\}$ would put σ and τ in a block, and ϵ in a different block. This case cannot occur due to the requirement that the blocks are connected. One can now go through the four cases, and observe that for each $Z = (Z_1, Z_2, Z_3)$, Z can be produced in exactly one of the four cases in the algorithm, because each step is distinguished by how the partitions they produce separate $\{\sigma, \epsilon, \tau\}$. Moreover, because within each case the X_i and Y_i can be recovered by taking appropriate intersections of the Z_i with G_1 and G_2, there is a unique pair of partitions of G_1 and G_2 that produce each (Z_1, Z_2, Z_3). Thus, $F(Z_1, Z_2, Z_3) = 1$. We now go through the cases in more detail:

- The case $X_3 = \emptyset$ and $Y_3 = \emptyset$, equivalently, $\sigma, \tau \in Z_1$. See Figure B.1(a).
  Recovery: $X_1 = Z_1 \cap G_1$, $X_2 = Z_2 \cap G_1$, $X_3 = \emptyset$, and $Y_1 = Z_1 \cap G_2$, $Y_2 = Z_2 \cap G_2$, $Y_3 = \emptyset$.
- The case $X_3 = \emptyset$, $Y_3 \neq \emptyset$, equivalently, $\sigma, \epsilon \in Z_1$, $\tau \in Z_3$. See Figure B.1(b).
  Recovery: $X_1 = Z_1 \cap G_1$, $X_2 = Z_2 \cap G_1$, $X_3 = \emptyset$, and $Y_1 = Z_1 \cap G_2$, $Y_2 = Z_2 \cap G_2$ $Y_3 = Z_3$.
- The case $X_3 \neq \emptyset$, $Y_3 = \emptyset$, equivalently, $\sigma \in Z_1$, $\epsilon, \tau \in Z_3$. See Figure B.1(c).
  Recovery: $X_1 = Z_1 \cap G_1$, $X_2 = Z_2 \cap G_1$, $X_3 = Z_3 \cap G_1$, and $Y_1 = Z_3 \cap G_2$, $Y_2 = Z_2 \cap G_2$, $Y_3 = \emptyset$.
- The case $X_3 \neq \emptyset$, $Y_3 \neq \emptyset$, equivalently, σ, ϵ, τ are each in a different block. See Figure B.1(d).
  Recovery: $X_1 = Z_1 \cap G_1$, $X_2 = \emptyset$, $X_3 = Z_2 \cap G_1$, and $Y_1 = Z_2 \cap G_2$, $Y_2 = \emptyset$, $Y_3 = Z_3 \cap G_2$.

Finally, we examine the relationship between Algorithm B.2 and Algorithm B.1. In particular, we will couple them together by sorting the elements of $\tilde{X}((G_1, w_1)$ so that those with the same sequence of weights appear together, and the same for $\tilde{X}((G_2, w_2))$. The weights of the two 3-partitions we are merging uniquely determines which of the four cases Algorithm B.2 is in since, by construction of the E coordinate of every node weight, a set is empty iff its weight is zero. The weights of the the two merging partitions also determine the weights of the resulting 3-partition, say (c_1, c_2, c_3). Thus, while the set level algorithm Algorithm B.2 putters through $\Theta(m_1 m_2)$ set-operations and $m_1 m_2$ updates to a function, the algorithm Algorithm B.1 computes $m_1 m_2$ and adds that that to the value of f over some efficiently computable tuple (c_1, c_2, c_3).

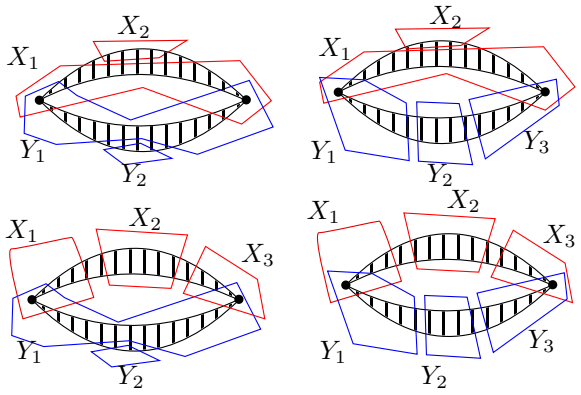
Finally, after having verified that Algorithm B.1 has the correct output, we remark that it consists of a single loop over the product of two sets, which has size bounded by $O(w(G_1)^2 w(G_2)^2)$, since the number of elements in $X(G)$ is in general bounded by $w(G)^2$. Within each loop, each $m_i \leq 2^{3|V(G)|}$, so the cost of multiplications and additions are polynomial in $|G|$. This concludes the proof. ∎

We next explain how to compute $X((G_1, w_1) \parallel (G_2, w_2))$ from $X((G_1, w_1))$ and $X((G_2, w_2))$.

Algorithm B.3 `ParallelPartitions`

Input: $X((G_1, w_1))$ and $X((G_2, w_2))$ for (G_1, w_1) and (G_2, w_2)
Output: $X((G_1, w_1) \parallel (G_2, w_2))$.

Set f to be constantly zero on $\{(a_1, a_2, a_3) : 0 \le a_i, \sum a_i = w((G_1, w_1) \parallel (G_2, w_2))\}$
for $(a_1, a_2, a_3, m_1) \in X(G_1, w_1)$ and $(b_1, b_2, b_3, m_2) \in X(G_2, w_2)$ **do**
  if $a_3 = 0$ AND $b_3 = 0$ AND ($a_2 = 0$ OR $b_2 = 0$) **then**
    $f(a_1 + b_1, a_2 + b_2, 0)$ += $m_1 m_2$
  if $a_3 = 0$ AND $b_3 \neq 0$ AND ($a_2 = 0$ OR $b_2 = 0$) **then**
    $f(a_1 + b_1 + b_3, a_2 + b_2, 0)$ += $m_1 m_2$
  if $a_3 \neq 0$ AND $b_3 = 0$ AND ($a_2 = 0$ OR $b_2 = 0$) **then**
    $f(a_1 + b_1 + a_3, a_2 + b_2, 0)$ += $m_1 m_2$
  if $a_3 \neq 0$ AND $b_3 \neq 0$ AND ($a_2 = 0$ OR $b_2 = 0$) **then**
    $f(a_1 + b_1, a_2 + b_2, a_3 + b_3)$ += $m_1 m_2$
Return f



(a) Case 1 (b) Case 2

(c) Case 3 (d) Case 4

Figure B.2: The four cases in Algorithm B.3

PROPOSITION B.22. *If (G, w) is admissible, then Algorithm B.3 runs correctly and in time polynomial in $(|G|, n(w(G)))$.*

Proof. This proof follows the same structure as Proposition B.21. We verify that the function F returned by Algorithm B.4 is the indicator of $\tilde{X}(G)$ in $\mathcal{P}(V(G))^3$. First, it is straightforward to check that $\mathrm{supp}(F) \subseteq \tilde{X}(G)$. Next, let $(Z_1, Z_2, Z_3) \in \tilde{X}(G)$. There are four cases based on how Z connects σ and τ.
1. Z_1 has a path through G_1 and G_2 from σ to τ.
2. Z_1 has a path through only G_1 from σ to τ.
3. Z_1 has a path through only G_2 from σ to τ.
4. No paths, that is: $\sigma \in Z_1$ and $\tau \in Z_3$.

One can now observe that for each $Z = (Z_1, Z_2, Z_3)$, Z can be produced in exactly one of the four cases in Algorithm B.4, because each case is distinguished what kind of paths in Z_1 there are from σ to τ. Moreover, because the X_i and Y_i can be recovered by taking appropriate intersections of the Z_i with G_1 and G_2, there is a unique pair of partitions of G_1 and G_2 that produce each (Z_1, Z_2, Z_3). Thus, $F(Z_1, Z_2, Z_3) = 1$. We now list the cases of the algorithm in more detail:

- The case $X_3 = \emptyset$ and $Y_3 = \emptyset$, equivalently, Z_1 has a path through G_1 and G_2 from σ to τ. See Figure B.2(a). Recovery: $X_1 = Z_1 \cap G_1$, $X_2 = Z_2 \cap G_1$, $X_3 = \emptyset$, and $Y_1 = Z_1 \cap G_2$, $Y_2 = Z_2 \cap G_2$, $Y_3 = \emptyset$.

Algorithm B.4 `SetLevelParallelPartitions`

Input: series-parallel graphs G_1 and G_2 along with $\widetilde{X}(G_1)$ and $\widetilde{X}(G_2)$

Output: $\widetilde{X}(G_1 \,\|\, G_2)$. (Convention to align with pictures G_1 will be the top SP-graph, whose partitions are denoted with X_i.)

Initialize F as the zero function on $\mathcal{P}(V(G))^3$
for $(X_1, X_2, X_3) \in \widetilde{X}(G_1)$ and $(Y_1, Y_2, Y_3) \in \widetilde{X}(G_2)$ **do**
  if $X_3 = \emptyset$ and $Y_3 = \emptyset$ AND ($X_2 = \emptyset$ or $Y_2 = \emptyset$) **then**
    $F(X_1 \cup Y_1, X_2 \cup Y_2, \emptyset)$ += 1
  if $X_3 = \emptyset$ and $Y_3 \neq \emptyset$ AND ($X_2 = \emptyset$ or $Y_2 = \emptyset$) **then**
    $F(X_1 \cup X_3 \cup Y_1, X_2 \cup Y_2, \emptyset)$ += 1
  if $X_3 \neq \emptyset$ and $Y_3 = \emptyset$ AND ($X_2 = \emptyset$ or $Y_2 = \emptyset$) **then**
    $F(X_1 \cup Y_1 \cup Y_3, X_2 \cup Y_2, \emptyset)$ += 1
  if $X_3 \neq \emptyset$ and $Y_3 \neq \emptyset$ AND ($X_2 = \emptyset$ or $Y_2 = \emptyset$) **then**
    $F(X_1 \cup Y_1, X_2 \cup Y_2, X_3 \cup Y_3)$ += 1
Return F

- The case $X_3 = \emptyset$ and $Y_3 \neq \emptyset$, equivalently, Z_1 has a path through only G_1 from σ to τ. See Figure B.2(b). Recovery: $X_1 = Z_1 \cap G_1$, $X_2 = Z_2 \cap G_1$, $X_3 = \emptyset$, and Y_1 the component of $Z_1 \cap G_2$ containing σ, $Y_2 = Z_2 \cap G_2$ and Y_3 is the component of $Z_1 \cap G_2$ containing τ.
- The case $X_3 \neq \emptyset$ and $Y_3 = \emptyset$, equivalently, Z_1 has a path through only G_2 from σ to τ. See Figure B.2(c). Recovery: X_1 is the component of $Z_1 \cap G_1$ containing σ, $X_2 = Z_2 \cap G_1$, X_3 is the component of $Z_3 \cap G_1$ containing τ, and $Y_1 = Z_1 \cap G_2$, $Y_2 = Z_2 \cap G_2$, $Y_3 = \emptyset$.
- The case $X_3 \neq \emptyset$ and $Y_3 \neq \emptyset$, equivalently, $\sigma \in Z_1$ and $\tau \in Z_3$. See Figure B.2(d). Recovery: $X_1 = Z_1 \cap G_1$, $X_2 = Z_2 \cap G_1$, $X_3 = Z_3 \cap G_1$, and $Y_1 = Z_1 \cap G_2$, $Y_2 = Z_2 \cap G_2$ and $Y_3 = Z_3 \cap G_2$.

The relationship between Algorithm B.4 and Algorithm B.3 is the same as in Proposition B.21, proving that Algorithm B.3 has the correct output. Moreover, it consists of a single loop over the product of two sets, which has size bounded by $O(w(G_1)^2 w(G_2)^2)$, since the number of elements in $X(G)$ is in general bounded by $w(G)^2$. Within each loop, each $m_i \leq 2^{3|V(G)|}$, so the cost of multiplications and additions are polynomial in $|G|$. □

THEOREM B.23. *Let (G, w) be a node $\mathbb{N}$-weighted series-parallel graph. Then $|P_2^0(G, w)|$ can be calculated in time polynomial in $(|G|, w(G))$.*[20]

Proof. We extend w to weights valued in $N = E \times \mathbb{N}$, by setting $w'(a) = (\neg\emptyset, w(a))$ for all $a \in V(G)$. Thus,(G, w') is admissible. We let T be a binary SP-tree for G. This is a binary tree with $|E(G)|$ leaves, so it has $O(|E(G)|)$ nodes. Each node of T is associated with a subgraph of G, and we make them into node-weighted series-parallel graphs by setting the E component to be $\neg\emptyset$ on all nodes, and by setting the $\mathbb{N}$ component of the weight function in any way that adds up correctly using Definition B.18; for example, if H is a node in T, with left child L and right child R, we can assign the $\mathbb{N}$ part of the weight on L to be the restriction of w to L, and on R to be the restriction of w to $R \setminus L$, and zero elsewhere. The resulting node-weighted graphs are all admissible by construction.

Moreover, the graph at each P node is the node-weighted parallel composition of the graphs at each child node, and the graph at each S node is the node-weighted series-composition of the graphs at each child node. Computing $X((H, w'))$ at each of the leaves takes time $O(N(w(G)))$. Computing the value of $X((H', w'))$ for each P or S node, given the values at the children, takes time $O(p(|G|, w(G)))$ for some polynomial fixed p (given by Proposition B.21 and Proposition B.22). Thus, the total time to compute $X((H', w'))$ at each node of the tree by memoization is $O(|E(G)|p(|G|, w(G)))$.

From $X((G, w'))$ we can calculate $|P_2^0(G, w)|$ as

$$|P_2^0(G, w)| = X(G, w')((\frac{w(V)}{2}, \neg\emptyset), (\frac{w(V)}{2}, \neg\emptyset), 0) + X(G, w')((\frac{w(V)}{2}, \neg\emptyset), 0, (\frac{w(V)}{2}, \neg\emptyset))$$

[20] The input to the polynomial is the size of $w(G)$, not the binary encoding of $w(G)$.

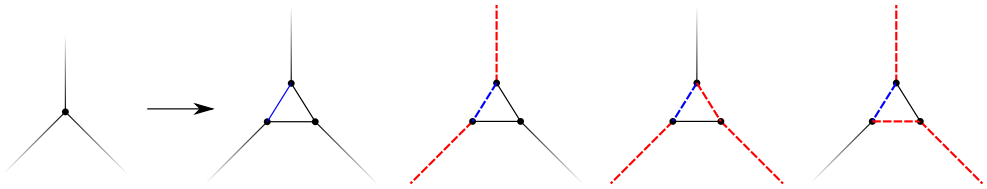


Figure B.3: a)Replacing each vertex of G with a triangle. Here the blue edge is the one added to J. b) The routing rules illustrating that any Hamiltonian cycle in G gives an extension of J to a simple cycle in G', and any extension of J in G' gives a Hamiltonian cycle of G.

B.5. The natural p-relation for simple cycles and un-ordered 2-partitions is not self-reducible. Here we prove that a natural encoding of the simple cycles of a graph is not self-reducible, unless P = NP. We take our inspiration from [66], where it is proven that a particular p-relation encoding 4 coloring a planar graph is not self-reducible. We use the same definition for self-reducibility as in [66].

We encode a graph $G = (V, E)$ in such a way that the edges are ordered. A solution $X \in SC(G)$ is described by an element of $2^{|E|}$, a binary sequence of length $|E|$, where the kth term is 1 if and only if the kth edge of G is in X. We call this p-relation R_{SC}.

Let Σ be some alphabet, where Σ has some ordering, and we extend that ordering to Σ^n for all n using the lexicographic order. We let $R \subseteq \Sigma^* \times \Sigma^*$ be any p-relation with $|y| = p(|x|)$ for all $(x, y) \in R$ for some polynomial $p(n)$. Now we define the following problem (see also [52, Definition 3.2]):

LF-R (Lexicographically First)
Input: $x \in \Sigma^*$
Output: The lexicographically first element of $R(x)$, provided $R(x) \neq \emptyset$.

For example, LF-R_{SC} is the problem of finding the lexicographically first simple cycle under the particular encoding of R_{SC}. The following proposition is well known:

Proposition B.24 ([66], Prefix-Search). *Suppose that R is a self-reducible p-relation, and suppose that there is a polynomial time Turing machine which, given x, answers if $R(x) \neq \emptyset$. Then there is a polynomial time algorithm for LF-R.*

The idea in [66] is to show that a certain p-relation R is not self-reducible, as long as P $\neq$ NP, by showing that checking $R(X) \neq \emptyset$ is in P, but that LF-R is NP-hard. We will follow the same approach by reducing to LF-R_{SC} from the following problem, which we will shortly show is NP-complete.

ExtendingToSimpleCycle
Input: An undirected graph G and a set of edges $J \subseteq E(G)$.
Output: YES if J can be extended to a simple cycle of G. NO, otherwise.

Proposition B.25. *`ExtendingToSimpleCycle` is NP-complete on the class of $3CCP$ graphs with face degree bounded by* 531.

Proof. Let $G \in \mathscr{C}_{177}$. Construct G' by replacing each vertex of G with a triangle as in Figure B.3a); the result remains $3CCP$ by Lemma A.1, and has face degree bounded by 531. Build a set J by taking one edge from each of those triangles. By examining the local routings in figure Figure B.3b), we can see that J has an extension to a simple cycle of G' iff G has a Hamiltonian cycle. Since the latter problem is NP-complete by Theorem 2.25, the proposition follows.

Theorem B.26. *Fix any class of graphs K which contains $\mathscr{C}_{531}$. Then the relation R_{SC} is not self-reducible on K assuming* P $\neq$ NP.

Proof. Let G, J be some instance of `ExtendingToSimpleCycle`. We order the edges of G so that J are the first edges. If there is an extension of J to a simple cycle, then one of the simple cycles extending J is the lexicographically first simple cycle among all simple cycles of G. If we assume that R_{SC} is self-reducible, then Proposition B.24 guarantees that we can determine the lexicographically first simple cycle in polynomial time, which puts the extension problem into P. This contradicts P $\neq$ NP because we have proven that `ExtendingToSimpleCycle` is NP-hard. □

We remark that, since we have proven Theorem B.26 in the context of plane graphs, we obtain results about the non-self reducibility of encodings of connected 2-partitions. One encoding that immediately reduces to the theorem just proven is to encode a connected 2-partition as the set of cut edges.