Fewer colors for perfect simulation of proper colorings

Mark Huber
Claremont McKenna College
mhuber@cmc.edu

Abstract

Given a graph $G$ and color set $\{1, \ldots, k\}$, a proper coloring is an assignment of a color to each vertex of $G$ such that no two vertices connected by an edge are given the same color. The problem of drawing a proper coloring exactly uniformly from the set of proper colorings is well-studied. Most recently, Bhandari and Chakraborty developed a polynomial expected time randomized algorithm for obtaining such draws when $k > 3\Delta$, where $\Delta$ is the maximum degree of the graph. Their approach used a bounding chain together with the coupling from the past protocol. Here a new randomized algorithm is presented based upon the randomness recycler protocol introduced by the author and Fill at FOCS 2000. Given $n$ vertices, this method takes $O(n \log(n))$ expected steps when $k > 2.27(\Delta - 1)$ for all $\Delta \geq 2$.

1 Introduction

Let $G = (V, E)$ be an undirected graph, and $C = \{1, \ldots, k\}$ be a set of colors. Then a coloring of the graph is $x \in V^C$, so each vertex $v \in V$ is labeled with $x(v) \in C$. A proper coloring is a coloring such that for all $\{v, w\} \in E$, $x(v) \neq x(w)$.

Let $\Delta$ be the maximum degree of the vertices of $G$. Then when $k \geq \Delta$, it is always possible to obtain a proper coloring of $G$. The problem of counting the number of proper colorings is #P-complete [13]. Using the reducible approximation scheme of Jerrum, Valiant, and Vazirani [15], it is possible to obtain an $(\epsilon, \delta)$-randomized approximation scheme by drawing uniformly from the set of possible colorings.

Early approaches to this problem concentrated on ways of approximately uniformly sampling from the set of proper colorings. Such an algorithm could return draws that came from within some total variation distance of uniformity, but does not exactly match the target distribution. Following notation of [1], let $k^+(\Delta)$ denote the smallest integer such that for any total variation distance $\nu$ greater than 0, there exists an algorithm that for every graph with maximum degree $\Delta$, the algorithm generates output that is within total variation $\nu$ in time polynomial in the size of the graph $G$. All approximate sampling methods work by
constructing a Markov chain whose stationary distribution is uniform over the set of proper colorings, and then finding conditions under which the chain is rapidly mixing.

Jerrum [14] showed that it was possible to generate such an approximate sample in polynomial time when $k \geq 2\Delta$, making $k_+ \leq 2\Delta$. His approach used the standard Gibbs sampler Markov chain. Bubley and Dyer [3] later showed that this type of argument could be extended to a wide variety of problems using path coupling. By analyzing a chain that took larger moves (flipping the colors of clusters), Vigoda [17] developed an algorithm that showed $k_+ \leq (11/6)\Delta$. This was improved recently by Chen et. al [4] who showed that $k_+ \leq (11/6 - c)\Delta$ where $c$ is about $10^{-4}$.

If the graph is restricted in some way, then the algorithms can be improved. In [5], Dyer, Frieze, Hayes, and Vigoda gave an algorithm for graphs of girth at least 6 and large enough $\Delta$ where $k \geq 1.49\Delta$ gives a polynomial time algorithm.

### 1.1 Perfect sampling of proper colorings

The algorithms described so far only output approximate colorings, leaving open the question of how easy it is to generate exactly uniformly from the set of proper colorings. Algorithms which generate exactly from a distribution using a random number of steps are called perfect sampling or perfect simulation algorithms.

Several protocols exist for designing such perfect sampling algorithms [12]. The most well-known is coupling from the past (CFTP), created by Propp and Wilson [16] to transform certain Markov chain approaches into perfect simulation. Their CFTP protocol was applied to the problem of uniformly sampling colorings using the notion of a bounding chain, introduced independently by Huber [11, 9] and Häggström and Neander [8].

Again following the notation of [1], let $k_0(\Delta)$ be the smallest number such that there exists a perfect simulation algorithm that outputs uniformly over the proper colorings of a graph in expected time polynomial in the graph size whenever $k \geq k_0(\Delta)$. Then in [10], it was shown that $k_0 \leq \Delta(\Delta + 1)$. Others [6] developed new algorithms for this problem, but it was not until recently that the $\Delta^2$ barrier was broken for general graphs.

Bhandari and Chakraborty [1] improved the original bounding chain approach of [11] to create a CFTP based algorithm that runs in polynomial expected time when $k \geq 3\Delta$. Their method works by building in a more complex coupling between different paths of the Markov chain.

In this work we introduce a different approach, based on the randomness recycler protocol created by Fill and the author [7, 10]. In this protocol, there is both a state $x$ and an index $x^*$. The index is over distributions $\{\pi(\alpha)\}$, and the invariant will hold throughout that $x \sim \pi(x^*)$. There is a target index $t^*$ such that $\pi(t^*)$ is the target distribution, namely, uniform over the proper colorings of the graph.

**The Randomness Recycler** Begin with an index $x^*$ such that is easy to sample $x \sim \pi(x^*)$ exactly. Then, at each step, given a state and index $(x^*, x)$, propose moving to a new state $y$ by making a random change in $x$. This state $y$ is then possibly accepted as coming from
a new distribution indexed by \( y^* \), which is closer (in a sense to be given later) to \( t^* \) than \( x^* \) is. If the state \( y \) is rejected from \( y^* \), then we move to the rejected distribution \( r^* \).

The result is a random walk both on states and on distributions, with the invariant that the state is an exact draw from the distribution given by the index holds at each step. This continues until the index reaches \( t^* \), as which point the state is output as a draw from the target distribution.

Here this protocol is applied to build an algorithm for sampling uniformly from the set of proper colorings. When \( k \) is large enough, the index is more likely to move towards \( t^* \) than away from it, leading to a polynomial expected time algorithm.

In Section 2, the new algorithm is presented, and shown to be a valid randomness recycler type algorithm for this problem. The main result is Theorem 1 in Section 3, which for fixed \( \Delta \) and \( k > 227(\Delta - 1) \), gives an algorithm for generating exactly from the set of \( k \)-colorings of an \( n \) node graph in \( O(n \ln(n)) \) expected time.

## 2 Description and correctness of the algorithm

This section both describes and proves the correctness of the new algorithm. Begin by setting up notation.

### 2.1 Notation

The algorithm keeps a pair \((x^*, x)\) consisting of an index and a state at all times. The state \( x \) is a coloring of \( V \) using color set \( \{0, 1, \ldots, k\} \). Refer to the color 0 as clear.

The index \( x^* \) is a coloring of \( V \) using color set \( \{-k, \ldots, -1, 0, 1, \ldots, k, k+1\} \). This coloring breaks the nodes \( V \) into three different types based on whether \( x^*(v) \leq 0 \), \( x^*(v) \in \{1, \ldots, k\} \), or if \( x^*(v) = k+1 \).

1. If \( x^*(v) = k+1 \), then \( x(v) \) is of unrestricted type.
2. If \( x^*(v) \in \{0, 1, \ldots, k\} \), then node \( v \) is frozen, and \( x(v) = x^*(v) \).
3. If \( x^*(v) < 0 \), then color \( |x^*(v)| \) is forbidden, and \( x(v) \neq |x^*(v)| \).

The frozen and forbidden nodes are formally defined as logical conditions as follows.

\[
p_1(x^*, x) = (\forall v \in V)((0 \leq x^*(v) \leq k) \rightarrow (x(v) = x^*(v))),
\]

\[
p_2(x^*, x) = (\forall v \in V)((x^*(v) < 0) \rightarrow (x(v) \neq -x^*(v))).
\]

Let

\[
N(v) = \{w \in V : \{v, w\} \in E\}
\]

be the neighbor set of \( v \). Let \( \Omega_0 \) be all colorings in \( V^{\bigcup_{\{0\}}} \) where two nodes of color 0 are allowed to be next to each other. That is,

\[
\Omega_0 = \{x \in V^{\bigcup_{\{0\}}} : (\forall v \in V)(\forall w \in N(v)((x(v) > 0) \rightarrow (x(w) \neq x(v)))).
\]
Then for an index $x^*$, let $\Omega(x^*)$ be elements of $\Omega_0$ where the frozen and forbidden conditions are both met. That is,

$$\Omega(x^*) = \{x \in \Omega_0 : p_1(x^*, x) \wedge p_2(x^*, x)\}$$

Let $\pi(x^*)$ denote the distribution that is uniform over $\Omega(x^*)$. Note that $\Omega((k+1, \ldots, k+1))$ is our target distribution, and $\Omega((0,0, \ldots, 0))$ only has a single state $x = (0, \ldots, 0)$, and so is easy to sample from.

In a randomness recycler algorithm, from the current state $(x^*, x)$ it is proposed to move to state $(y^*, y)$. This state is accepted with probability $a(x^*, x, y^*, y)$. If the proposed state is not accepted, the state changes to $(z^*, z)$, called the recycled state. Next consider how this general procedure can be applied to the proper coloring problem.

### 2.2 The Algorithm

A randomness recycler algorithm operates by randomly changing the state in order to move the index $x^*$ closer to the target index $(k+1, \ldots, k+1)$. Unfortunately, sometimes the state rejects the move, in which case the index typically moves farther away from the target index. This random walk continues until the target index is reached, at which point the algorithm terminates.

For the proper coloring problem, the new randomness recycler type algorithm is as follows. At each step of the algorithm, one restriction is attempted to be removed. These restrictions are removed in the following order: two nodes frozen at a positive color, node forbidden a color adjacent to a frozen node of the same color, forbidden node, unique frozen node at a positive color, node frozen at color 0.

1. **Start** with $x^* = x = (0,0, \ldots, 0)$.

2. **While** $x^* \neq (k+1, k+1, \ldots, k+1)$, take one step. Execute one of the following steps, where if two or more conditions hold, execute the one lowest in alphabetical order.

   - (a) **Nodes** $\{v_1, v_2\}$ **frozen at** $c \in C$. Let $v$ be either $v_1$ or $v_2$ so that removing $v$ from the set of nodes frozen at 0 leaves the set of nodes frozen at 0 connected. Forbid all unrestricted neighbors of $v$ from having color $c$, and refreeze $v$ at 0.

   - (b) **Node** $v$ **forbidden color** $c$ **adjacent to node** $w$ **frozen at color** $c$. Remove the forbidden condition from $v$, changing it to unrestricted.

   - (c) **Node** $v$ **forbidden color** $c$. Choose a new color for $v$ uniformly from $C \setminus x(N(v))$. Draw $U$ uniformly from $[0,1]$. If color $c \notin x(N(v))$, or $c \in x(N(v))$ and $U \leq 1 - 1/\#(x(N(v)))$, accept the change and set $v$ to unrestricted. Otherwise, reject and execute recycle step that searches for a neighbor of color $c$ that is given below.

   - (d) **Node** $v$ **frozen at** $c \in C$. Randomly permute all the color labels, change $v$ to unrestricted.
(e) **There exist nodes frozen at 0** Let \( v \) be a node frozen at 0 such that its removal would leave the set of nodes frozen at 0 connected. Draw new color \( c \) uniformly from \( C \) for \( v \). If the result is a proper coloring, change \( v \) to unrestricted. Otherwise, reject and execute the recycle step that searches for a neighbor of color \( c \).

3. **Output** \( x \).

Under conditions (c) and (e), the algorithm might need to recycle the state by searching for a neighbor with a particular color. This can be done as follows.

**Recycle step (search for neighbor with color \( c \))** Search uniformly at random among the unrestricted neighbors of \( v \). As each neighbor \( w_i \) is examined, if \( x(w_i) \neq c \), change \( w \) to forbidden color \( c \). When the neighbor \( w_\ell \) is found with \( x(w_\ell) = c \), freeze \( w_\ell \) at color \( c \). Finally, freeze \( v \) at color 0.

### 2.3 Invariants

Several invariants are maintained throughout these steps. The formal proofs that these hold are all by induction: show that the invariant holds at the start, and then show that for each of (a) through (e), the steps (including possibly the recycle step) maintain the invariant. Because they all have the same form, the proofs are deferred to the appendix.

**Lemma 1.** The number of nodes frozen at a color in \( C \) is either 0, 1, or 2.

**Lemma 2.** All nodes forbidden a color are forbidden the same color \( c \). Any nodes frozen at a color in \( C \) are also frozen at color \( c \).

**Lemma 3.** Any node that is forbidden a color or frozen at a color in \( C \) is adjacent to at least one node frozen at 0.

### 2.4 Finding the acceptance probabilities

Suppose that from state \((x^*, x)\) where \( x \sim \pi(x^*) \), the state \( y \) is randomly generated, and the hope is to accept \( y \) as a draw from \( \pi(y^*) \). The chance of accepting is determined in a way similar to that of the Metropolis-Hastings algorithm. Let \( a(x^*, x, y^*, y) \) be the chance of accepting a proposal \( y \) as a draw from \( y^* \).

For a given state \( x \in \Omega(x^*) \), let \( w(x) \propto (P)(X = x) \) where \( X \sim \pi(x^*) \). Let \( p(x^*, x, y) \) be the probability of proposing \( y \) from state \((x^*, x)\). Then the weight of state \( y \) is

\[
w(y) = \sum_{x: p(x^*, x, y) > 0} w(x)p(x^*, x, y)a(x^*, x, y^*, z).
\]

For uniform distributions, choose \( w(x) = 1 \). The goal is to make \( w(y) \) represent a uniform distribution over \( \Omega(y^*) \), so it should be a constant (over \( y \)) for \( y \in \Omega(y^*) \) and 0 otherwise.
Definition 1. Say that an acceptance procedure is valid if \( a(x^*, x, y^*, y) \) is chosen so that

\[
w(y) = \sum_x w(x)p(x^*, x, y)a(x^*, x, y^*, y)
\]

yields weights for state \( y \) that are proportional to the target distribution \( y \sim \pi(y^*) \).

Note that for either (a), (b), or (d), acceptance is guaranteed.

Acceptance for condition (d) Begin by considering condition (d) in the algorithm. If \( y \) is the state resulting from a random permutation of the colors of \( C \), then \( y(v) \) is now the color that \( x(v) \) was mapped to. Their are then \((k - 1)!\) different possible \( x \) values that each have a \( 1/k! \) chance of moving to \( y \). Hence

\[
w(y) = (k - 1)! \frac{1}{k!} a(x^*, x, y^*, y),
\]

and so we can make \( a(x^*, x, y^*, y) = 1 \) and have \( w(y) \) not depend on \( y \). Therefore, the output of this step can be accepted with probability 1, and there is never a need to reject and recycle. This shows the following lemma.

Lemma 4. Setting \( a(x^*, x, y^*, y) = 1 \) in condition (d) yields a valid acceptance procedure.

Acceptance for (b) In condition (b), \( y(v) = 0 \), and any unrestricted neighbors of \( v \) are forbidden to have color \( x^*(v) \). Because the value of \( x(v) \) is frozen at \( x^*(v) \), there is exactly one state \( x \) that leads to \( y \). This step leads to \( y \) with probability 1. Hence

\[
w(y) = a(x^*, x, y^*, 1),
\]

and making \( a(x^*, x, y^*, y) = 1 \) makes this acceptance valid.

Acceptance for (a) Again every state \( y \in \Omega(y^*) \) has exactly one \( x \) that leads to it, and so the acceptance rate of 1 ensures that \( y \sim \pi(y^*) \).

Unlike conditions (a), (b), and (d), under conditions (c) and (e) acceptance is not guaranteed.

Acceptance for (c) Consider condition (c). Here \( y^* \) is the same as \( x^* \), except \( y^*(v) = k + 1 \). Given \( y \), \( x(V \setminus v) \) is determined, but the \( x(v) \) value is not. It can be any color not already occupied by a neighbor, except the forbidden color \( x^*(v) \). That means that the number of \( x \) with positive \( p(x^*, x, y) \) depends on the colors of the neighbors of \( v \).

Let \( k_v = |C \setminus x(N(v))| \). If a neighbor of \( v \) has the forbidden color, then the number of such \( x \) is \( k_v \), and

\[
w(y|\{x^*(v)| \in x(N(v))\}) = k_v \frac{1}{k_v} a(x^*, x, y^*, y) = a(x^*, x, y^*, y).
\]
If a neighbor of $v$ does not have the forbidden color, then the number of such $x$ is $k_v - 1$, making
\[ w(y|x^*(v) \notin x(N(v))) = (k_v - 1) \frac{1}{k_v} a(x^*, x, y^*, y). \]

**Lemma 5.** In Step (c), set
\[ a(x^*, x, y^*, y) = \begin{cases} \frac{(k_v - 1)}{k_v} & |x^*(v)| \in x(N(v)) \\ 1 & |x^*(v)| \notin x(N(v)) \end{cases} \]
This is a valid acceptance procedure.

**Proof.** Whether or not $|x^*(v)| \in x(N(v))$, this makes $w(y) = \frac{(k_v - 1)}{k_v}$, for $y \in \Omega(y^*)$. Therefore it is a valid acceptance procedure.

**Acceptance for (e)** Finally, under condition (e) it holds that $y^*$ is the same as $x^*$ except $y^*(v) = k + 1$ whereas $x^*(v) = 0$. Because $x(v)$ is frozen at 0, there is exactly one $x$ that has positive probability of moving to the proposed state $y$, and $p(x^*, x, y) = 1/k$. Hence all elements of $\Omega(y^*)$ have equal weight. That means we accept with probability 1 if $y \in \Omega(y^*)$ and probability 0 if $y \notin \Omega(y^*)$.

**Lemma 6.** In (e), set
\[ a(x^*, x, y^*, y) = \begin{cases} 1 & y \in \Omega(y^*) \\ 0 & y \notin \Omega(y^*) \end{cases} \]
Then this is a valid acceptance procedure.

**Proof.** By only accepting when $y \in \Omega(y^*)$, we have $w(y) = 1/k$ for $y \in \Omega(y^*)$ and $w(y) = 0$ for $y \notin \Omega(y^*)$. Hence the weight for $y$ is valid for the uniform distribution over $\Omega(y^*)$.

That completes the determination of the acceptance probabilities.

### 2.5 Recycling

When the pair $(y^*, y)$ is rejected as the new state, a move is made to $(z^*, z)$, where $z$ attempts to recycle as much of the state $y$ as possible. Let $z = f(x^*, x, y)$. Unlike $y$, which has a weight factor of $a(x^*, x, y^*, y)$, $z$ has a weight factor of $(1 - a(x^*, x, y^*, y))$ since rejection occurred. Moreover, it is necessary to consider multiple $y$ states that could lead to the same $z$. This makes
\[ w(z) = \sum_x \sum_{y: f(x^*, x, y) = z} w(x)p(x^*, x, y)(1 - a(x^*, x, y^*, y)). \]

As with the acceptance probabilities, there is also a notion of valid for rejection and recycling. Let $f(x^*, y^*, y) = z$ give the map to the reject state.

**Definition 2.** Say that a reject and recycle procedure is valid if $w(z)$ yields weights for $z$ such that when normalized they give $z \sim \pi(z^*)$. 

7
Recycling for condition (c) Rejecting in (c) means that instead of gaining a factor of \( a(x^*, x, y^*, y) \), the weight of state \( y \) gains a factor of \( 1 - a(x^*, x, y^*, y) \). Therefore, given that rejection (event \( A^C \)) occurs, the weight of \( y \) in Step 2 is
\[
\begin{align*}
w(y \mid x^*(v) \in x(N(v))) &= 1/k_v \\
w(y \mid x^*(v) \notin x(N(v))) &= 0
\end{align*}
\]

Also, when rejection occurs, it occurs because there must be a neighbor of \( v \) with the forbidden color \( |x^*(v)| \), and the weight \( 1/k_v \) depends on the state \( y \) over \( N(v) \).

To solve the unequal weights and neighbor of color \( |x^*(v)| \), do two things. First, freeze the node \( v \) at color 0, this will solve the weighting problem.0 Second, search the neighbors of \( v \) in a random order until the neighbor with the forbidden color is found. Any other neighbor that was searched must then be forbidden to have that color as well. Freeze the neighbor that was found to have the forbidden color. Finally, freeze the color of the original vertex \( v \) at 0.

Lemma 7. With the above recycling procedure for (c), the weights are valid.

Proof. When rejection occurs, there must exist an unrestricted \( w \) (a neighbor of \( v \)) such that the \( w \) has the forbidden color for \( v \). That means that the number of different choices of \( x(v) \) that lead to \( y \) and then \( z \) is exactly \( k_v \).

The chance of moving from \( x \) to \( y \), rejecting, and then moving to \( z \) involves first picking the color \( y(v) \), then randomly choosing the unrestricted neighbors \( \{w_1, \ldots, w_{n-1}\} \) to examine right before randomly finding \( w_n \) with the forbidden color. Let \( d_u(v) \) denote the number of unrestricted neighbors of \( v \) in \( x^* \). Then the weight associated with \( z \) under \( z^* \) is
\[
w(z) = k_v \frac{1}{k_v} \cdot \frac{(i-1)!}{d_u(v)(d_u(v) - 1) \cdots (d_u(v) - (n-1))}.
\]
Note that the right hand side is a constant entirely determined by \( z^* \). Therefore with rejection the weight is uniform over \( z^* \).

3 Analysis

To analyze the expected number of steps taken by the algorithm, consider the step where trying to unfreeze a node frozen at 0 leads to forbidden nodes, which in turn can lead to more nodes frozen at 0. The goal is to show that the average number of frozen 0 nodes after (e) is executed is smaller than 1. Let \( R_0 \) denote this replication number. (This type of analysis is similar to that of path coupling for showing that the Markov chain is rapidly mixing [2], and also used for infectious diseases.)

When that holds, each frozen node decays away exponentially fast, and the total number of steps will decay away in time \( O(\ln(1/R_0)) \). Because the algorithm begins with \( n = #(V) \)
nodes frozen at 0, the result is an algorithm that runs in $O(n \ln(n))$ expected time, with a constant that depends on $\gamma$.

To set notation, let $R_0$ denote the expected number of nodes frozen at 0 that arise from (e). Let $R_1$ denote the expected number of nodes frozen at 0 that arise from (c).

To bound $R_0$ and $R_1$, first bound the probability of rejection in (c) and in (e).

**Lemma 8.** The probability of rejection in (c) is at most

$$f(k, \Delta) = \frac{1}{k - (\Delta - 1)} \left[ 1 - \left( 1 - \frac{1}{k - (\Delta - 1)} \right)^{\Delta - 1} \right].$$

**Proof.** In order to reject a forbidden node in step (c), there must be a neighbor of $v$ that has the forbidden color, and then an event of probability $1/\#(x(N(v))) \leq 1/(k - (\Delta - 1))$ must occur.

For each neighbor $w$ of $v$, think about taking a step in the Gibbs sampler. There is at most a $1/(k - \#(x(N(w)))) \leq 1/(k - (\Delta - 1))$ chance that the color becomes the forbidden color. Hence there is at least a

$$\left[ 1 - \frac{1}{k - (\Delta - 1)} \right]^{\Delta - 1}$$

chance that no neighbors have the forbidden color, and factoring in the chance of accepting even with a neighbor with the forbidden color completes the proof. \qed

**Lemma 9.** The probability of rejection in (e) is at most $(\Delta - 1)/k$.

**Proof.** Rejection occurs when the color chosen for the node frozen at 0 matches one of the unrestricted neighbors. There are at most $\Delta - 1$ such neighbors, and the chance of hitting that neighbor’s color is at most $1/k$. \qed

**Lemma 10.** For $f(k, \Delta)$ as in Lemma 8, for a node frozen at 0 adjacent to another node frozen at 0,

$$R_0 \leq \frac{\Delta - 1}{k} \cdot \left[ 1 + \frac{f(k, \Delta)(\Delta - 1/2)}{1 - (3/2)f(k, \Delta)(\Delta - 1/2)} \right].$$

**Proof.** Consider first (e) applied to a node frozen at 0 that is adjacent to another node frozen at 0. If acceptance occurs in (e), then the node frozen at 0 is immediately removed. Therefore, it only replicates if rejection occurs. In this case, the original node remains, and on average $(\Delta/2) - 1$ forbidden nodes are created, and one node that is frozen. This frozen node will eventually be removed by (d) once the forbidden nodes are all dealt with. Hence

$$R_0 \leq \mathbb{P}(\text{reject in (e)})[1 + ((\Delta/2) - 1)R_1].$$

Since $\mathbb{P}(\text{reject in (e)}) \leq (\Delta - 1)/k$,

$$R_0 \leq (\Delta - 1)k^{-1}[1 + ((\Delta/2) - 1)R_1].$$
Condition (b) removes forbidden nodes directly, so only reduces $R_1$. If (c) rejects, then a frozen node, $(\Delta/2) - 1$ forbidden nodes (on average) and a node frozen at the forbidden color will be created. Then in the next step (a) will turn the frozen node into 1 node frozen at 0 and at most $\Delta - 1$ forbidden nodes. Hence

$$R_1 \leq \mathbb{P}(\text{reject in (c)})[2 + ((3/2)\Delta - 2)R_1].$$

Using $\mathbb{P}(\text{reject in (c)}) \leq f(k, \Delta)$ gives

$$R_1 \leq \frac{2f(k, \Delta)}{1 - f(k, \Delta)((3/2)\Delta - 2)},$$

which gives

$$R_0 \leq \frac{\Delta - 1}{k} \cdot \left[1 + \frac{f(k, \Delta)(\Delta - 1/2)}{1 - f(k, \Delta)((3/2)\Delta - 2)}\right].$$

A small table of some values of $k$, $\Delta$, and our upper bound on $R_0$.

| $k$ | $\Delta$ | Upper bound on $R_0$ |
|-----|-----------|----------------------|
| 3   | 2         | 0.5                  |
| 5   | 3         | 0.7448275862         |
| 7   | 4         | 0.9424603175         |
| 44  | 20        | 0.8911424929         |
| 89  | 40        | 0.9516872879         |

Since $R_0 < 1$ for each $(k, \Delta)$ pair above, the algorithm runs in $O(n \ln(n))$ for these pairs. For each value of $k$ above, it holds that $k \geq 2.37(\Delta - 1)$. In fact, this is sufficient to guarantee that $R_0 < 1$.

Note that if the algorithm gets down to one node frozen at 0, then there is at least a $1/k$ chance of accepting the draw for $k \geq \Delta + 1$. Therefore, as long as the expected time to reach one node frozen at 0 is $O(g(k, \Delta))$, the expected time to completely generate the graph will be $O(kg(k, \Delta))$.

**Theorem 1.** For $k \geq 2.37(\Delta - 1)$, where $\Delta \geq 2$, the algorithm generates a perfect coloring of a graph with $n$ nodes in $O(\Delta n \ln(n))$ expected number of steps.

**Proof.** From the discussion earlier it suffices to show that $k \geq 2.37(\Delta - 1)$ implies $R_0 < 1$. This can be verified directly using the previous lemma for $\Delta \in \{2, \ldots, 17\}$. For larger values of $\Delta$, upper bounds on $f(k, \Delta)$ are needed.

For $k > 2.27(\Delta - 1)$, let $x = 1/[(\Delta - 1)/k - 1] > 1/1.27$. So $x < 0.7875$. For $y \in (0, 0.24]$, $1 - y \geq \exp(-y - 0.6y^2)$.

This gives

$$f(k, \Delta) \leq \frac{x}{\Delta - 1}(1 - \exp(-x - 0.6x^2/(\Delta - 1))).$$
Since our bound on $R_0$ is increasing in $f(k, \Delta)$,

$$R_0 \leq \frac{1}{2.27} \left[ 1 + \frac{x(\Delta - 1)^{-1}(1 - \exp(-x - 0.6x^2/(\Delta - 1)))(\Delta - 1/2)}{1 - x(\Delta - 1)^{-1}(1 - \exp(-x - 0.6x^2/(\Delta - 1)))[(3/2)(\Delta - 1) - 1/2]} \right]$$

$$\leq \frac{1}{2.27} \left[ 1 + \frac{x(1 - \exp(-x - 0.6x^2/(\Delta - 1)))(3/2)(\Delta - 1) - 1/2)}{1 - (3/2)x(1 - \exp(-x - 0.6x^2/(\Delta - 1))} \right].$$

The right hand side is decreasing in $\Delta$, so replacing with $\Delta = 19$ makes it as large as possible. Similarly, it is increasing in $x$, so replacing $x$ with its lower bound of $1/1.27$ makes this as large as possible. The result is

$$R_0 \leq 0.999,$$

for all $\Delta \geq 18$, which completes the proof.

\section*{A Proofs of Invariants}

\textit{Proof of Lemma 1.} At the start of the algorithm, the number of nodes frozen at a color in $C$ is 0. One of (a) through (e) is executed at each step. If there are 2 nodes frozen at a color in $C$, then the number is reduced to 1 by (a). Otherwise, each step increases the number of such nodes by at most 1. Therefore the invariant holds.

\textit{Proof of Lemma 2.} Suppose there are no node forbidden or frozen at a color in $C$. In (e), attempts to remove a frozen 0 condition result in the creation of a frozen node and forbidden node of the same color. In (a) and (d) a frozen node of color $c$ is removed, possibly creating nodes forbidden color $c$. In (b) a forbidden condition is possibly removed, and in (d) a frozen condition is removed. In (c) attempts to remove a forbidden color $c$ might possible result (through the recycle step) in new nodes forbidden $c$ or frozen at $c$. Either way, the invariant is maintained.

\textit{Proof of Lemma 3.} At the start there are no forbidden nodes. Such nodes are created at the recycle step of (c) and (e). In the recycle step the forbidden nodes created are all neighbors of the original node, which is frozen at color 0. In (a), forbidden nodes are created adjacent to a frozen node which is then refrozen at color 0 as well. Hence forbidden nodes are always adjacent to frozen 0 nodes. Similarly, nodes frozen at a positive color are also only created next to nodes frozen at 0.

\section*{References}

[1] S. Bhandari and S. Chakraborty. Improved bounds for perfect sampling of $k$-colorings in graphs. 2019.

[2] R. Bubley and M. Dyer. Path coupling: a technique for proving rapid mixing in Markov chains. In \textit{Proc. 38th Sympos. on Foundations of Computer Science}, pages 223–231, 1997.
[3] R. Bubley and M. Dyer. Path coupling, Dobrushin uniqueness, and approximate counting. School of Computer Studies Research Report Series 97.04, University of Leeds, 1997.

[4] Sitan Chen, Michelle Delcourt, Ankur Moitra, Guillem Perarnau, and Luke Postle. Improved bounds for randomly sampling colorings via linear programming. In Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 2216–2234. SIAM, 2019.

[5] Martin Dyer, Alan Frieze, Thomas P Hayes, and Eric Vigoda. Randomly coloring constant degree graphs. Random Structures & Algorithms, 43(2):181–200, 2013.

[6] Weiming Feng, Heng Guo, and Yitong Yin. Perfect sampling from spatial mixing. arXiv preprint arXiv:1907.06033, 2019.

[7] J. A. Fill and M. L. Huber. The Randomness Recycler: A new approach to perfect sampling. In Proc. 41st Sympos. on Foundations of Comp. Sci., pages 503–511, 2000.

[8] O. Häggström and K. Nelander. On exact simulation from Markov random fields using coupling from the past. Scand. J. Statist., 26(3):395–411, 1999.

[9] M. Huber. Perfect sampling using bounding chains. Annals of Applied Probability, 14(2):734–753, 2004.

[10] M. Huber. Approximation algorithms for the normalizing constant of Gibbs distributions. Ann. Appl. Probab., 51(1):92–105, 2015. arXiv:1206.2689.

[11] M. L. Huber. Exact sampling and approximate counting techniques. In Proc. 30th Sympos. on the Theory of Computing, pages 31–40, 1998.

[12] Mark L. Huber. Perfect Simulation. Number 148 in Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press, 2015.

[13] François Jaeger, Dirk L Vertigan, and Dominic JA Welsh. On the computational complexity of the jones and tutte polynomials. In Mathematical Proceedings of the Cambridge Philosophical Society, volume 108, pages 35–53. Cambridge University Press, 1990.

[14] M. Jerrum. A very simple algorithm for estimating the number of $k$-colourings of a low-degree graph. SIAM J. Comput., 22:1087–1116, 1995.

[15] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. Theoret. Comput. Sci., 43:169–188, 1986.

[16] J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Structures Algorithms, 9(1–2):223–252, 1996.
[17] Eric Vigoda. Improved bounds for sampling colorings. *Journal of Mathematical Physics*, 41(3):1555–1569, 2000.