A Study of the Entropy Compression Technique for Graph Coloring Problems

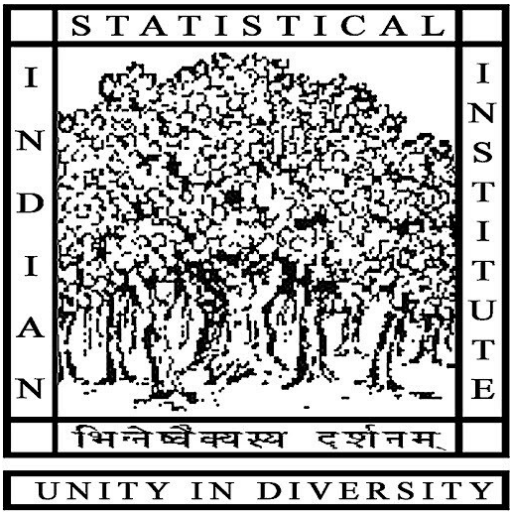
A dissertation submitted by

Jatin Kumar Dey
(ROLL NO. CS2216)

under the guidance of

Dr. Mathew C. Francis

in partial fulfillment of the requirements
for the award of the degree of
Master of Technology
in
Computer Science



INDIAN STATISTICAL INSTITUTE, KOLKATA

Declaration

I, **Jatin Kumar Dey**, hereby affirm that this report titled **A Study of the Entropy Com- pression Technique for Graph Coloring Problems**, submitted to the Indian Statistical Institute, Kolkata, as part of the requirements for the **Master of Technology in Computer Science**, is my original work. This research was conducted under the guidance of **Dr. Mathew C. Francis** and has not been previously submitted for the award of any degree or diploma at any other institution or university. I have diligently followed academic ethics and integrity throughout this project. All external sources of information, statements, or results have been appropriately cited and acknowledged.

Jatin Kumar Dey
MTCS2216
Indian Statistical Institute
Kolkata - 700108
June 12, 2024

THESIS CERTIFICATE

This is to certify that the thesis titled **A Study of the Entropy Compression Technique for Graph Coloring Problems**, submitted by **Jatin Kumar Dey (CS2216)**, to the Indian Statistical Institute, Kolkata for the award of the degree of **Master of Technology in Computer Science**, is a *bona fide* record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Mathew C. Francis
Computer Science Unit
ISI Chennai Centre
Date: June 12, 2024

ACKNOWLEDGEMENTS

I am profoundly grateful to my thesis advisor, Professor Mathew C. Francis, for his unwavering support, guidance, and insightful discussions, which were instrumental in the successful completion of this thesis. I deeply appreciate his invaluable mentorship. I extend my sincere thanks to all the esteemed professors at the Indian Statistical Institute (ISI) for generously imparting their vast knowledge and expertise in Computer Science. Their teachings and suggestions significantly enriched my research. I would like to express my heartfelt appreciation to my mother, Bani Dey, and brother, Sukanta Dey, for their unwavering faith in me. Their encouragement led me to this esteemed institution. I am forever grateful for the exceptional education I received at ISI. Lastly, I wish to express my gratitude to all my classmates and friends for their unwavering support and encouragement throughout my Master of Technology (MTech) journey. Their presence has been a source of motivation. I also extend my thanks to any individuals inadvertently omitted from this acknowledgment.

Jatin Kumar Dey
CS2216
ISI Kolkata
Date: June 12, 2024

Contents

Abstract

This thesis explores the application of the entropy compression technique to various graph coloring problems, offering an innovative approach to addressing significant challenges in graph theory. Entropy compression, particularly the Moser-Tardos [12] framework, transforms probabilistic existence proofs into explicit, constructive algorithms, thereby enhancing our understanding and expanding the toolkit available for solving these challenges.

Graph coloring problems involve assigning colors to the vertices or edges of a graph under specific constraints, such as ensuring no two adjacent vertices or edges share the same color. These problems are both theoretically rich and practically significant, with applications in scheduling, register allocation in compilers, and network frequency assignment. Motivated by the work of Esperet and Parreau [6], this research focuses on the acyclic edge chromatic number. Through rigorous analysis and algorithmic design, this study demonstrates how entropy compression has the potential to improve existing bounds to complex combinatorial problems.

Introduction

Graph coloring problems are a fundamental area of study in combinatorics and computer science. They involve assigning colors to the vertices or edges of a graph under specific constraints, such as ensuring no two adjacent vertices (or edges) share the same color. These problems are theoretically rich and have practical applications in areas like scheduling, register allocation in compilers, and network frequency assignment. One of the pivotal techniques in this domain has been the Lovász Local Lemma (LLL), a powerful probabilistic tool that provides a way to prove the existence of certain combinatorial structures. The LLL has been instrumental in establishing bounds for various combinatorial constructs, including diagonal Ramsey numbers and edge colorings in graphs. The essence of the LLL lies in its ability to handle events with limited dependencies, making it an invaluable tool for demonstrating the existence of rare configurations that meet specific criteria. However, the LLL is inherently non-constructive; it shows that certain objects exist but does not provide a direct method to find them, which often limits its direct application in algorithm design.

Entropy Compression:

Entropy compression is a method that transforms probabilistic existence proofs into explicit, constructive algorithms. This thesis explores how this technique can be effectively applied to various graph coloring problems, enhancing our understanding and expanding the methods available for solving these issues. Moser's entropy compression argument represents a significant advancement in addressing this limitation. This method offers a constructive approach that converts probabilistic existence proofs into explicit algorithms [13]. This approach has proven particularly effective in problems like acyclic edge coloring, where it not only establishes bounds but also guarantees the existence of valid colorings through a deterministic process. Building on this foundation, Esperet and Parreau [6] (2013) applied a deterministic variant of Moser's entropy compression [12] to enhance the bounds on the acyclic chromatic index, illustrating the method's potential for improving existing results in graph theory and beyond.

Acyclic Edge Chromatic Number:

The acyclic edge chromatic number, denoted $a'(G)$, of a graph G is the minimum number of colors needed to color the edges of G such that no two adjacent edges share the same color and no cycle in G is bichromatic. This concept is important for ensuring efficient and conflict-free resource allocation in various applications.

Esperet and Parreau's work:

Esperet and Parreau used the entropy compression method to show that every graph with a maximum degree Δ has an acyclic edge chromatic number $a'(G)$ of at most $4\Delta-4$. Their analysis hinges on a simple procedure that systematically recolors edges to eliminate bichromatic cycles. The method involves recording the history of corrections (and sometimes the steps taken to resolve conflicts). In our research, we placed a greater emphasis on understanding the mechanics of the entropy compression method rather than focusing on counting precisely the records produced by the algorithm. By applying this method to various other problems, we aimed to gain deeper insights into its underlying principles and operational dynamics. This approach allowed us to become more familiar with the method and also exploring its practical applications. By experimenting with different combinatorial problems, we were able to observe how the method performs in diverse scenarios, thus providing a comprehensive understanding of what the entropy compression technique fundamentally does and how it achieves its results. At this time we are also not focusing on the running time of the algorithm, rather we are interested to the analysis of the algorithm and how the entropy compression argument gives the results involving some particular properties of the graph used as the input of algorithm.

Using straightforward counting and less constraints in the algorithm, we initially tried to showcase how the method actually works and we got the result that $a'(G) \leq 6\Delta - 5$. This involved a basic application of the entropy compression technique, where the algorithm iteratively attempts to recolor edges while avoiding the creation of bichromatic cycles. By counting in a more concise way, we further enhanced the bound on the acyclic chromatic index to $4.9\Delta - 5$.

In more detail, our initial approach applied standard combinatorial arguments combined with the entropy compression method to establish that the acyclic chromatic index $a'(G)$ for a graph G with maximum degree Δ satisfies $a'(G) \leq 6\Delta-5$. The algorithm used in this approach iteratively selects edges to recolor in a way that prevents the formation of bichromatic cycles. At each step, the algorithm records the sequence of changes and the history of the recoloring process grows.

To achieve the enhanced bound of $4.9\Delta-5$, we refined our counting method a little bit more precise for the analysis of the combinatorial objects related to the encoding of the algorithm's history.

However, Esperet and Parreau went further by employing one more constraint in the algorithm and more sophisticated counting techniques involving Dyck words and rooted plane trees. Dyck words are binary strings that represent valid sequences of matched parentheses, and they can be used to model certain types of paths in graphs. Rooted plane trees are trees embedded in the plane such that no two edges cross, which can be used to represent hierarchical structures in graphs. By counting records more precisely using these combinatorial objects, they improved the bound to $4\Delta - 4$.

Diagonal Ramsey numbers:

For diagonal Ramsey numbers, denoted $R(s, s)$, the goal is to determine the smallest number n such that any graph of order n will contain either a clique or an independent set of size s. This problem is a classic example of a combinatorial problem where the existence of certain structures is guaranteed by probabilistic methods, but finding those structures constructively is challenging.

Traditionally, the Erdős probabilistic method has established that $R(s, s) > 2^{s/2}$. This method relies on the probabilistic construction of graphs and shows that, with high probability, a randomly chosen graph will not contain a clique or independent set of the specified size, thereby proving the lower bound. However, this approach does not provide a constructive way to find such graphs.

We applied the entropy compression method to this problem in two different ways:

Basic Entropy Compression (ECC(G, S)): This approach reproduces the Erdős bound by employing a simple entropy compression technique. Here, the algorithm iteratively attempts to construct a graph while avoiding the formation of cliques or independent sets of size s. The method ensures termination by demonstrating that the entropy increases faster than the information content, guaranteeing $R(s, s) > 2^{s/2}$.

Modified Entropy Compression (MECC(G, S)): This refined approach incorporates elements from the Lovász Local Lemma to enhance the bounds further. By carefully analyzing the dependencies between edges and using a more sophisticated compression scheme, we achieve the same bound given by LLL i.e. $R(k, k) > n$ if $e \cdot 2^{1-\binom{k}{2}} \cdot \binom{k}{2}\binom{n-2}{k-2} < 1$. This method not only provides a more rigorous bound but also retains the constructive nature of the entropy compression argument.

The deterministic entropy compression method is also adapted to k-uniform hyperedge coloring and the k-SAT problem. In this problem, the goal is to color the edges of a hypergraph (a generalization of a graph where edges can connect more than two vertices) such that no monochromatic hyperedges exist. By applying the entropy compression method, we aim to establish bounds on the minimum number of colors required for such a coloring and guarantee the existence of valid colorings.

*****

Acyclic Edge Chromatic Number

In graph theory, edge coloring is a fundamental concept where edges of a graph are assigned colors such that no two adjacent edges share the same color. A more restrictive and intriguing variation of this concept is the acyclic edge coloring, where the colored graph must not only avoid adjacent edges having the same color but also ensure that no cycle in the graph is bichromatic (i.e., it must not contain a cycle with only two colors).

The acyclic edge chromatic number, denoted as $a'(G)$, is the minimum number of colors required to achieve an acyclic edge coloring of a graph G. This parameter extends the classical edge chromatic number, or chromatic index, by incorporating an additional layer of complexity through the acyclic condition.

Formally, for an integer $k \geq 2$, a proper edge k-coloring of a graph $G = (V, E)$ is a mapping

$$c : E(G) \to \{1, 2, \ldots, k\}$$

such that for any two adjacent edges e_1 and e_2, $c(e_1) \neq c(e_2)$. A graph G is edge k-colorable if it has a proper edge k-coloring. The chromatic index $\chi'(G)$ is the smallest k such that G is edge k-colorable.

An *acyclic edge coloring* is a proper edge coloring with the additional property that no cycle in the graph is bichromatic. In other words, for every cycle in the graph, at least three distinct colors must appear among its edges.

The *acyclic edge chromatic number* $a'(G)$ of a graph G is the smallest number of colors needed to achieve an acyclic edge coloring of G. Formally,

$$a'(G) = \min\{k \mid \exists \text{ an acyclic edge coloring } c : E \to \{1, 2, \ldots, k\}\}.$$

Observe, in any proper edge coloring there cannot be any odd length cycle which is bichromatic.

Various bounds on $a'(G)$

For a graph G with maximum degree $\Delta(G) = 1$, it is trivial that $a'(G) = 1$. For $\Delta(G) = 2$, we have $a'(G) = 2$ if G is a linear forest and $a'(G) = 3$ if G contains a cycle. For the graphs with maximum degree $\Delta(G) = 3$, Fiamčík [8] [9] proved that $a'(K_4) = a'(K_{3,3}) = 5$. If $\Delta(G) = 3$ and G is connected and $G \neq K_4$, $G \neq K_{3,3}$, then $a'(G) < 4$. He then further investigated the classes of graphs with $\Delta(G) = 3$ for which $a'(G) = 3$ or $a'(G) = 4$. For other general graphs

with $\Delta(G) > 4$, only some bounds are known. For the lower bound of $a'(G)$, the inequality

$$\Delta(G) < \chi'(G) < a'(G)$$

trivially holds, where $\chi'(G)$ is the edge chromatic number of a graph G satisfying Vizing's inequality [17]:

$$\Delta(G) < \chi'(G) < \Delta(G) + 1,$$

Conjecture 1 (Fiamčík [7], and Alon, Sudakov, and Zaks [1])**.** *For every graph G, $a'(G) \leq \Delta + 2$.*

This conjecture has been verified up to now only for $\Delta = 1$, 2, and 3. Later Guldan [10] showed that there exists a large class of graphs with $a'(G) \geq \Delta + 2$.

Theorem 1 (Guldan [10])**.** *Let k be a positive integer, and let $G = (V(G), E(G))$ be a graph with maximum degree $\Delta(G) = \Delta$, $|V(G)| = 2k$, and $|E(G)| > (\Delta + 1)(k - 1) + 1$. Then $a'(G) > \Delta + 2$.*

Proof. By contradiction. Assume G has an acyclic regular edge coloring with $\Delta + 1$ colors. Let $L_1, L_2, \ldots, L_{\Delta+1}$ be the monochromatic sets of edges of this coloring. For any $i \neq j$, it must hold that $|L_i \cup L_j| < 2k$, otherwise $L_i \cup L_j$ would contain a cycle, contradicting the acyclic property. From this, it follows that for the maximum number of colored edges, we have:

$$\sum_{i=1}^{\Delta+1} |L_i| < k + (k-1)\Delta = (\Delta + 1)(k - 1) + 1.$$

This is a contradiction to the assumption $|E(G)| > (\Delta + 1)(k - 1) + 1$, so $\Delta + 1$ colors cannot suffice. Thus, $a'(G) \geq \Delta + 2$. From the theorem, it follows that there exists a large class of graphs with $a'(G) \geq \Delta(G) + 2$. □

The study of acyclic edge coloring has a rich history, marked by significant contributions from eminent researchers in the field. In 1991, Alon et al. [14] introduced the application of the Lovász Local Lemma to establish bounds on the acyclic chromatic index $a'(G)$, demonstrating that $a'(G) \leq 64\Delta$, where Δ denotes the maximum degree of the graph G. This seminal work laid the foundation for subsequent advancements in acyclic edge coloring. Building upon this result, Molloy and Reed (1998) [11] further refined the bound to 16Δ, showcasing the continuous evolution of techniques in this area. In 2011, Ndreca et al. [15] enhanced the bound to $\lceil 9.62(\Delta - 1) \rceil$ by leveraging a stronger version of the Lovász Local Lemma developed by Bissacot et al. [2] These milestones underscore the significance and ongoing progress in the study of acyclic edge coloring.

We first show how Esperet and Parreau [6] use entropy compression to show that every graph with maximum degree Δ has an acyclic edge chromatic number $a'(G)$ of at most $6\Delta - 6$, and then show how this bound can be enhanced to $4.9\Delta - 5$. The bound results from the analysis of a very simple procedure using the so-called entropy compression method.

The algorithm and its analysis

Let k be a positive integer. We describe an algorithm that takes a graph G and an input string s whose symbols are from the alphabet $\{1, 2, \ldots, k - 2\Delta + 2\}$ and uses it to try to find an acyclic edge coloring of G using the colors $\{1, 2, \ldots, k\}$. Let $G = (V, E)$, the maximum degree of G be Δ, $|V| = n$ and $|E| = m$. The algorithm first fixes an ordering $\prec$ on the vertices of G and an ordering $\prec'$ on the edges of G. In the beginning, all edges of G are uncolored. Repeat the following steps until either they have been executed $|s|$ times or there are no more uncolored edges. In the i-th iteration, where $1 \leq i \leq |s|$, do the following steps:

- Take the uncolored edge that is smallest in the ordering $\prec'$. Let us denote this edge as e_i.
- Assign e_i the $s[i]$-th smallest color from the set of colors that do not appear on any edge adjacent to e_i.
- If this assignment creates a 2-colored cycle, then uncolor e_i and all the other edges in this cycle except two of them (we will understand it later).

Note that at the end of each iteration, the colors that have been assigned to the edges of G form a partial acyclic edge-coloring of G; i.e. an assignment of colors to a subset of edges in such a way that no two adjacent edges get the same color, and there is no 2-colored cycle.

Algorithm 1 : Acyclic Edge Coloring Algorithm

procedure AEC(G, s) ▷ Input: G and a string $s \in \{1, 2, \ldots, k - 2\Delta + 2\}^*$
  Fix an ordering $\prec$ on $V(G)$ and an ordering $\prec'$ on $E(G)$
  Initialize the color of every edge to 0
  $i \leftarrow 0$
  while G is not fully colored **and** $i < |s|$ **do**
    $i \leftarrow i + 1$
    Let e_j be the first uncolored edge in the edge ordering $\prec'$
    Let S be the set of colors on the edges incident with e_j
    Color e_j with the $s[i]$-th smallest color in $\{1, 2, \ldots, k\} \setminus S$
      ▷ Since there are at most $2(\Delta - 1)$ edges that are adjacent to e_j, $|\{1, 2, \ldots, k\} \setminus S| \geq k - 2\Delta + 2$. So such a color always exists.
    Output "0"
    if a bicolored cycle C of length $2l$ is present in the graph **then**
      ▷ Note that bicolored cycles are of even length and since none existed before this step, C contains e_j
      Let $e_j = uv$, where $u \prec v$
      Let $h_1, h_2, \ldots, h_{2l}$ be the edges of C listed in the cyclic order in which $h_1 = uv$, and v is the common vertex between h_1 and h_2
      Uncolor all edges of C except the h_2 and h_3
      Output $x \in \{1, 2, \ldots, \Delta - 1\}^{2l-2}$ ▷ How to compute x will be described later
    end if
  end while
  Output for each edge $e \in E(G)$ the color assigned to it; this is a string in $\{0, 1, 2, \ldots, k\}^m$.
end procedure

Let $s \in \{1, 2, \ldots, k - 2\Delta + 2\}^*$ such that $|s| = t$. Let $f(s)$ be the string that is printed by the algorithm AEC(G, s). Suppose that the graph G does not get fully colored by the algorithm AEC(G, s). Then the main loop of the algorithm gets executed exactly t times, and therefore the algorithm prints exactly t zeroes before the final step in which it outputs the partial coloring. Suppose that the algorithm detects the formation of a 2-colored cycle r times. For $j \in \{1, 2, \ldots, r\}$, let x_j be the string that is output by the j-th invocation of line 17 of the algorithm (i.e., just after detecting the j-th 2-colored cycle).

Let Φ be the string corresponding to the partial coloring that is output by the algorithm in its final step. Then the output string $f(s)$ has the form

$$f(s) = 00 \ldots 0x_1 00 \ldots 0x_2 0 \cdots 0x_r 00 \ldots 0\Phi$$

where for each $j \in \{1, 2, \ldots, r\}$, we have that $|x_j|$ is even, and each block of consecutive zeroes has length at least 1, except the last block of zeroes, which may not necessarily exist.

Encoding of an x_j, for $j \in \{1, 2, \ldots, r\}$:

Let C_j be the j-th bicolored cycle detected by the algorithm. Let C_j be of length $2l$. Also let us assume that C_j was detected during the i-th iteration of the main loop of the algorithm. As mentioned before, it is clear that e_i is an edge in the cycle C_j. The string x_j is an encoding of the cycle C_j such that $|x_j| = 2l - 2$. Let $e_i = uv$, where $u \prec v$. Let $h_1, h_2, \ldots, h_{2l}$ be the edges of C_j listed in the cyclic order where $h_1 = uv$, and v is the common vertex between h_1 and h_2. Let z_p be the common vertex for the edges h_p and h_{p+1} (where the indices are cyclic and from the set $\{1, 2, \ldots, 2l\}$). For each $p \in \{1, 2, \ldots, 2l\}$, let E_p be the edges that are incident on z_p and we define y_p to be the position of the edge h_{p+1} in the ordering $\prec'$ restricted to $E_p \setminus \{h_p\}$. We define x_j to be the string $y_1 y_2 \ldots y_{2l-2}$. Since $|E_p| \leq \Delta$ for each $p \in \{1, 2, \ldots, 2l\}$, we have that $y_p \in \{1, 2, \ldots, \Delta - 1\}$. Thus $x_j \in \{1, 2, \ldots, \Delta - 1\}^{2l-2}$.

The partial edge coloring Φ:

The partial edge coloring Φ of G that is output at the final step is a string of length m where $\Phi[i]$ is color of the i-th edge in the edge ordering $\prec'$, where the symbol 0 is used to denote the fact that the corresponding edge is uncolored. Thus $\Phi \in \{0, 1, \ldots, k\}^m$. Hence Φ is one of $(k + 1)^m$ possible strings of length m.

Blocks of $f(s)$:

Let $block(i)$ be the substring of $f(s)$ that is printed by the algorithm in the i-th iteration of its main loop. Thus $f(s)$ is of the form $block(1)block(2) \ldots block(t)\Phi$. One can think of $block(i)$ as a record of the actions performed by the algorithm in the i-th iteration of its main loop. Note that given $f(s)$, the strings $block(1), block(2), \ldots, block(t)$ can be determined easily: every iteration that prints non-zero symbols prints a 0 followed by a string of non-zero symbols, and every other iteration prints a single 0. Thus, for each $i \in \{1, 2, \ldots, t\}$, the string $block(i)$ is either 0 or a string of non-zero symbols prefixed with a 0, and these can be derived easily

from the string $f(s)$.

For $i \in \{1, 2, \ldots, t\}$, let X_i represent the collection of uncolored edges after the i-th iteration of the main loop of the algorithm. We let X_0 be the set of uncolored edges of G at the beginning of the algorithm. Also, for $i \in \{1, 2, \ldots, t\}$, let Φ_i denote the partial coloring of the graph G at the end of the i-th iteration of the main loop. We will now demonstrate that $f(s)$ uniquely identifies s.

Lemma 1. *For each $i \in \{1, 2, \ldots, t\}$, X_i is uniquely determined by the output $f(s)$.*

Proof. Clearly, $X_0 = E(G)$. We claim that given X_{i-1} and $f(s)$, X_i is uniquely determined for each $i \in \{1, 2, 3, \ldots, t\}$. Note that this will complete the proof of the lemma. Notice that in the i-th iteration of the main loop, the algorithm chooses as e_i the smallest edge in the ordering $\prec'$ that is uncolored. Since X_{i-1} is the set of uncolored edges at the end of the $(i-1)$-th iteration of the loop, and also at the beginning of the i-th iteration of the loop, it is clear that $e_i = \min_{\prec'} X_{i-1}$. Thus, e_i is uniquely determined given $f(s)$ and X_{i-1}. If $block(i)$ is of the form "0", then $X_i = X_{i-1} \setminus \{e_i\}$, since e_i is the only edge that gets colored in the i-the iteration of the loop and no edges are uncolored (as otherwise $block(i)$ would not have been just 0). Since e_i is uniquely determined given $f(s)$ and X_{i-1}, we have that X_i is uniquely determined given $f(s)$ and X_{i-1}, and we are done in this case. Otherwise, $block(i) = 0x$, where $x \in \{1, 2, \ldots, \Delta - 1\}^{2l-2}$. This means that a bicolored cycle C was detected after the edge e_i was colored. From the way the string x is encoded, it can be seen that the cycle C is uniquely determined by the string x and the edge e_i. Let $h_1, h_2, \ldots, h_{2l}$ be the edges of C as defined in the algorithm. Note that the edges $h_1, h_2, h_3, \ldots, h_{2l}$ are uniquely determined by the string x and the edge e_i. Then $X_i = X_{i-1} \cup (E(C) \setminus \{h_2, h_3\})$, since every edge of C other than h_2 and h_3 get uncolored during this iteration of the algorithm, and no other edges are uncolored. This means that X_i is uniquely determined given $f(s)$ and X_{i-1}. □

Lemma 2. *For each $i \in \{1, 2, \ldots, t\}$, Φ_{i-1} and $s[i]$ are uniquely determined given $f(s)$.*

Proof. Clearly, $\Phi_t = \Phi$. We claim that given Φ_i and $f(s)$, Φ_{i-1} is uniquely determined for each $i \in \{1, 2, 3, \ldots, t\}$. Note that this will complete the proof of the lemma.

So using the previous lemma one can find e_i i.e. the edge colored in the i-th iteration of the main loop of the algorithm. Then the corresponding set of colors (S) of the adjacent edges of e_i can also be found. Now if $block(i) = 0$, that means e_i is the only edge that gets colored in the i-the iteration of the loop and no edges are uncolored (as otherwise $block(i)$ would not have been just 0), then one can find the corresponding entry for e_i in Φ_i according to the ordering in $\prec'$ and changing only that entry to 0 results Φ_{i-1}. To determine $s[i]$, find the position of $color(e_i)$ from the elements of the set $\{1, 2, \ldots, k\} \setminus S$ arranged in ascending order, say p. So $s[i] = p$. Otherwise, $block(i) = 0x$, where $x \in \{1, 2, \ldots, \Delta - 1\}^{2l-2}$. This means that a bicolored cycle C was detected after the edge e_i was colored in i-th iteration. From the way the string x is encoded, it can be seen that the cycle C is uniquely determined by the string x and the edge e_t. Let $h_1, h_2, \ldots, h_{2l}$ be the edges of C as defined in the algorithm. Note that

the edges $h_1, h_2, h_3, \ldots, h_{2l}$ are uniquely determined by the string x and the edge e_t. Since every edge of C other than h_2 and h_3 get uncolored during this iteration, so to get the colors of edges of $(E(C) \setminus \{h_2, h_3\})$, first one can find the corresponding colors for h_2 and h_3 from their corresponding entries in Φ_i, then coloring the edges $\{h_1, h_4, h_5, \ldots, h_{2l}\}$ alternatively using the colors of h_2 and h_3 results Φ_{i-1}. Then get the position of $color(h_3)$ from the elements of the set $\{1, 2, \ldots, k\} \setminus S$ arranged in ascending order, say p^*. So, here $s[i] = p^*$. □

Corollary 1. *Let s_1 and s_2 be distinct input strings such that neither AEC(G, s_1) nor AEC(G, s_2) colors the graph fully. Then $f(s_1) \neq f(s_2)$.*

Let $t \in \mathbb{N}$. Suppose that AEC(G, s) does not fully color the graph for any input string s of length t. Let Z be the set of input strings of length t — i.e $Z = \{1, 2, \ldots, k - 2\Delta + 2\}^t$. Hence, $|Z| = (k - 2\Delta + 2)^t$. Let $f(Z)$ be the set of output strings corresponding to input strings in Z; i.e. $f(Z) = \{f(s) \colon s \in Z\}$.

Let $s \in Z$. Removing Φ, i.e. last m characters of $f(s)$, get $f^*(s)$. Then by replacing each non-zero character with 1 in $f^*(s)$, we get $f^\bullet(s)$, as shown below:

$$\begin{aligned} f(s) &= 0000x_1 0000x_2 \cdots 0x_r 00 \underbrace{\qquad\qquad}_{\text{last } m \text{ characters}} \\ \Rightarrow f^*(s) &= 0000x_1 0000x_2 \cdots 0x_r 00 \\ \Rightarrow f^\bullet(s) &= 0000 \underbrace{11\ldots11}_{|x_1| \text{ many 1's}} 0000 \underbrace{11\ldots11}_{|x_2| \text{ many 1's}} 0 \underbrace{11\ldots11}_{|x_r| \text{ many 1's}} 00 \end{aligned}$$

Lemma 3. *For each $s \in Z$, $|f^\bullet(s)| \leq 2t$, implying that $|f^\bullet(Z)| \leq 2^{2t}$. Therefore, $|f^*(Z)| \leq 2^{2t}.(\Delta - 1)^t$.*

Proof. Observe that every 1 in $f^\bullet(s)$ corresponds to an action of uncoloring of some colored edge (we shall call this an "uncoloring step") that is done by the algorithm. Clearly, the *AEC algorithm* does the action of coloring an edge (a "coloring step") exactly t times. Because the number of uncoloring steps cannot be greater than the number of coloring steps, we have that the number of 1-s in $f^\bullet(s)$ is at most the number of 0-s. Thus, $|f^\bullet(s)| \leq 2t$. Since $f^\bullet(s)$ is a binary string, so $|f^\bullet(Z)| \leq 2^{2t}$. As each 1 in an $f^\bullet(s)$ corresponds to an element of $\{1, 2, \ldots, \Delta - 1\}$ in $f^*(s)$, we have $|f^*(Z)| \leq 2^{2t}.(\Delta - 1)^t$. □

Lemma 4. $|f(Z)| \leq (k + 1)^m \cdot 2^{2t} \cdot (\Delta - 1)^t$.

Proof. As Φ can take at most $(k + 1)^m$ different values,

$$|f(Z)| \leq (k + 1)^m \cdot |f^*(Z)| \leq (k + 1)^m \cdot 2^{2t} \cdot (\Delta - 1)^t.$$

□

Recall that we have assumed that the algorithm does not fully color the graph G for any string in Z. From Lemma 1, we know that $|f(Z)| \geq |Z|$. Thus:

$$(k + 1)^m \cdot 2^{2t} \cdot (\Delta - 1)^t \quad \geq \quad (k - 2\Delta + 2)^t$$

$$\begin{aligned} \Rightarrow (k+1)^m \cdot (4\Delta-4)^t &\geq (k-2\Delta+2)^t \\ \Rightarrow t &\leq \frac{m\log(k+1)}{\log(\frac{k-2\Delta+2}{4\Delta-4})} \end{aligned}$$

Thus, if the algorithm does not fully color the graph for any string of length t, then t must satisfy the above inequality. Notice that if we choose $k = 6\Delta - 5$, then the above inequality becomes

$$\begin{aligned} t &\leq \frac{m\log(6\Delta-4)}{\log(\frac{4\Delta-3}{4\Delta-4})} \\ \Rightarrow t &\leq \frac{m\log(6\Delta-4)}{\log(1+\frac{1}{4\Delta-4})} \end{aligned}$$

This means that if $k = 6\Delta - 5$, then for every integer $t > \frac{m\log(6\Delta-4)}{\log(1+\frac{1}{4\Delta-4})}$, the assumption that the algorithm does not fully color the graph G for any input string of length t must be false. In other words, for every integer $t > \frac{m\log(6\Delta-4)}{\log(1+\frac{1}{4\Delta-4})}$, there is some string of length t, which when given as input to the algorithm causes the graph G to be fully colored. This implies that there is some acyclic edge coloring of the graph G using at most $k = 6\Delta - 5$ colors, or in other words, $a'(G) \leq 6\Delta - 5$.

To use a tighter bound instead of $|f^\bullet(Z)| \leq 2^{2t}$, we use a slightly different approach. For each $s \in Z$, let $f^\circ(s)$ be the string obtained by replacing the substring "11" with "1" in $f^\bullet(s)$, as shown below:

$$f^\bullet(s) = 0000\ \underbrace{11\ldots11}_{|x_1| \text{ many 1's}}\ 0000\ \underbrace{11\ldots11}_{|x_2| \text{ many 1's}}\ 0\ \underbrace{11\ldots11}_{|x_r| \text{ many 1's}}\ 00$$

$$\Rightarrow f^\circ(s) = 0000\ \underbrace{11\ldots11}_{\frac{|x_1|}{2} \text{ many 1's}}\ 0000\ \underbrace{11\ldots11}_{\frac{|x_2|}{2} \text{ many 1's}}\ 0\ \underbrace{11\ldots11}_{\frac{|x_r|}{2} \text{ many 1's}}\ 00.$$

Recall that each of $|x_1|, |x_2|, \ldots, |x_t|$ is even. Note that $f^\bullet(s) = f^\bullet(s')$ if and only if $f^\circ(s) = f^\circ(s')$. Hence, there is a clear bijection between $f^\bullet(Z)$ and $f^\circ(Z)$. This implies that $|f^{bullet}(Z)| = |f^\circ(Z)|$.

Lemma 5. *For each $s \in Z$, $|f^\circ(s)| \leq \frac{3t}{2}$ and so, $|f^\circ(Z)| \leq 2^{\frac{3t}{2}}$.*

Proof. As we have seen that $|f^\bullet(s)| \leq 2t$, and the total number of 0-s is exactly t, we have that there are at most t 1-s in $f^\bullet(s)$. Since we construct $f^\circ(s)$ by replacing "11" with "1" in $f^\bullet(s)$, there are at most $\frac{t}{2}$ many 1-s in $f^\circ(s)$, which means that the length of $f^\circ(s)$ is at most $\frac{3t}{2}$. □

From the above lemma, it follows that $|f(Z)| \leq (k+1)^m \cdot |f^\bullet(Z)| \cdot (\Delta-1)^t = (k+1)^m \cdot |f^\circ(Z)| \cdot (\Delta-1)^t \leq (k+1)^m \cdot 2^{\frac{3t}{2}}(\Delta-1)^t$. Recall that from our assumption that the algorithm does not fully color the graph G for any input string in Z and Lemma 1, we know

that $|f(Z)| \geq |Z|$. Thus:

$$\begin{aligned} (k+1)^m \cdot 2^{\frac{3t}{2}} \cdot (\Delta-1)^t &\geq (k-2\Delta+2)^t \\ \Rightarrow (k+1)^m \cdot (2\sqrt{2}\Delta - 2\sqrt{2})^t &\geq (k-2\Delta+2)^t \\ \Rightarrow t &\leq \frac{m\log(k+1)}{\log(\frac{k-2\Delta+2}{2\sqrt{2}\Delta-2\sqrt{2}})} \end{aligned}$$

Thus, if the algorithm does not fully color the graph for any string of length t, then t must satisfy the above inequality. Notice that if we choose $k = (2+2\sqrt{2})\Delta - (1+2\sqrt{2})$, then the above inequality becomes

$$t \leq \frac{m\log(2+2\sqrt{2}(\Delta-1))}{\log(1+\frac{1}{2\sqrt{2}(\Delta-1)})}$$

This means that if $k = (2+2\sqrt{2})\Delta - (1+2\sqrt{2})$, then for every integer $t > \frac{m\log(2+2\sqrt{2}(\Delta-1))}{\log(1+\frac{1}{2\sqrt{2}(\Delta-1)})}$, the assumption that the algorithm does not fully color the graph G for any input string of length t must be false. In other words, for every integer $t > \frac{m\log(2+2\sqrt{2}(\Delta-1))}{\log(1+\frac{1}{2\sqrt{2}(\Delta-1)})}$, there is some string of length t, which when given as input to the algorithm causes the graph G to be fully colored. This implies that there is some acyclic edge coloring of the graph G using at most $k = (2+2\sqrt{2})\Delta - (1+2\sqrt{2}) \approx \lceil 4.9\Delta - 4 \rceil$ colors, or in other words, $a'(G) \leq \lceil 4.9\Delta - 4 \rceil$.

Discussion

From the above proof, it follows that for every graph G having maximum degree Δ, $a'(G) \leq \lceil 4.9\Delta - 4 \rceil$. Put in another way, suppose that there exists a graph G having maximum degree Δ such that $a'(G) > \lceil 4.9\Delta - 4 \rceil$. Let $k = (2+2\sqrt{2})\Delta - (1+2\sqrt{2})$. Then the algorithm AEC$(G, s)$, which tries to color the edges of G using k colors, will not succeed in fully coloring the graph G for any input string s. Let $t \in \mathbb{N}$ such that $t > \frac{m\log(2+2\sqrt{2}(\Delta-1))}{\log(1+\frac{1}{2\sqrt{2}(\Delta-1)})}$ and let Z be the set of all possible input strings of length t. From the proof above, we can see that the number of possible output strings produced by input strings of length t, i.e. $|f(Z)|$, is strictly smaller than $|Z|$, the number of input strings of length t. This means that the number of bits required to encode the input strings in Z is strictly more than the number of bits required to encode the output strings produced by the algorithm when given those input strings. Moreover, as we proved, the input strings are recoverable from the output strings. Thus our algorithm is an impossible compression algorithm — a contradiction.

*****

Diagonal Ramsey Number

The *diagonal Ramsey number* $R(k,k)$ is defined as the smallest positive integer n such that any complete graph on n vertices, with edges colored using two colors, contains a monochromatic k-clique — here, a monochromatic k-clique is a subset of k vertices such that all edges between those vertices have the same color. Formally, a k-edge coloring of a graph G is a mapping $c : E(G) \to S$, where $|S| = k$. Note that a k-edge coloring need not be a proper edge coloring. Then $R(k,k)$ is the smallest integer n such that for every 2-edge coloring $c : E(K_n) \to \{R, B\}$ of the complete graph K_n, there is a subgraph isomorphic to a K_k whose all edges are colored R, or all edges are colored B.

In other words,

$$R(k,k) = \min\{n \in \mathbb{N} \mid \forall c : E(K_n) \to \{R,B\}, \exists H \subseteq K_n \text{ that is isomorphic to } K_k$$

$$\text{such that } c(e) = R \quad \forall e \in E(H), \text{ or } c(e) = B \quad \forall e \in E(H)\}.$$

This definition emphasizes that within any sufficiently large complete graph, a certain degree of order is unavoidable, no matter how the edges are colored.

Theorem 2 (Erdős [5])**.** *For $k \geq 3$, $R(k,k) > 2^{k/2}$.*

Proof. Consider a random coloring of K_n where each edge is colored independently red or blue with probability 1/2. For any particular $S \subseteq V(K_n)$ such that $|S| = k$, the probability that the edges between pairs of vertices in S all receive the same color is $2 \cdot (\frac{1}{2})^{\binom{k}{2}} = 2^{1-\binom{k}{2}}$.

The number of subsets S of $V(K_n)$ of cardinality k is $\binom{n}{k}$, and therefore, using the union bound, the probability that there is at least one monochromatic k-clique is at most

$$\binom{n}{k} 2^{1-\binom{k}{2}}.$$

Now if $\binom{n}{k} \cdot 2^{1-\binom{k}{2}} < 1$, then there exists a coloring with no monochromatic k-clique. If $n < 2^{\frac{k}{2}}$ then we have:

$$\binom{n}{k} \cdot 2^{1-\binom{k}{2}} < \binom{2^{\frac{k}{2}}}{2} \cdot \left(\frac{1}{2}\right)^{\binom{k}{2}-1} < 2^{1-\frac{k^2+k}{2}} \cdot \frac{2^{\frac{k^2}{2}}}{k!} < 1$$

Hence, if $n < 2^{\frac{k}{2}}$, there exists a coloring with no monochromatic k-clique, proving that $R(k,k) \geq 2^{\frac{k}{2}}$. □

Frank Ramsey [16] first showed the existence of $R(k,k)$ for all $k \in \mathbb{N}$. Please refer to [3] for a survey of bounds on $R(k,k)$. In the next theorem, we derive an upper bound for $R(k,k)$. The statement of the theorem and its proof are similar to those that can be found in [4].

Theorem 3. *There is a monochromatic clique of size k in every coloring of the edges of K_n using two colors if $n \geq 2^{2k-1} - 1$; i.e. $R(k,k) \leq 2^{2k-1} - 1$.*

Proof. Consider a coloring of the edges of K_n using two colors $\{R, B\}$. We define $S_1 = V(K_n)$ and define v_1 to be an arbitrarily chosen vertex of S_1. We now define $v_2, v_3, \ldots, v_{2k-1}$ and $S_2, S_3, \ldots, S_{2k-1}$ inductively. We shall also ensure that for each $i \in \{2, 3, \ldots, 2k-1\}$, $|S_i| \geq 2^{2k-i-1} - 1$. Suppose that for some $i \in \{2, 3, \ldots, 2k-1\}$, v_{i-1} and S_{i-1} are defined, and that $|S_{i-1}| \geq 2^{2k-i} - 1$. We then define v_i and S_i as follows. Note that $|S_{i-1}| \geq 2^{2k-i} - 1$. Since $i \leq 2k-1$, this means that $|S_{i-1}| \geq 1$. We choose an arbitrary vertex of S_{i-1} as v_i. Note that there are $|S_{i-1}| - 1 \geq 2^{2k-i} - 2$ edges from v_i to other vertices in S_{i-1}. By Pigeonhole Principle, at least $\left\lceil \frac{2^{2k-i}-2}{2} \right\rceil = 2^{2k-i-1} - 1$ of these edges are of the same color, say 'R'. We define $S_i = \{u \in S_{i-1} \setminus \{v_i\} \mid uv_i \text{ is colored 'R'}\}$. Clearly, we have $|S_i| \geq 2^{2k-i-1} - 1$. Note that the edges from v_i to vertices in S_i all have the same color. Also note that it is possible that $S_{2k-1} = \emptyset$.

Notice for each $i \in \{1, 2, \ldots, 2k-2\}$, the edges $v_iv_{i+1}, v_iv_{i+2}, \ldots, v_iv_{2k-1}$ are all of the same color since $v_{i+1}, v_{i+2}, \ldots, v_{2k-1} \in S_i$ and the edges from v_i to each vertex in S_i have the same color. We say that the vertex v_i is of type 'R' if the edges $v_iv_{i+1}, v_iv_{i+2}, \ldots, v_iv_{2k-1}$ are all colored 'R'; otherwise, i.e. if the edges $v_iv_{i+1}, v_iv_{i+2}, \ldots, v_iv_{2k-1}$ are all colored 'B', we say that the vertex v_i is of type 'B'. Again by Pigeonhole Principle, we know that there is a subset $S \subseteq \{v_1, v_2, \ldots, v_{2k-2}\}$ such that $|S| \geq k-1$ and all vertices in S are of the same type. Now consider the set $S \cup \{v_{2k-1}\}$. Let us assume without loss of generality that each vertex in S is of type 'R'. Then we have by our observations above that each vertex $v_j \in S$, where $j \in \{1, 2, \ldots, 2k-2\}$, is connected to vertices in $(\{v_{j+1}, v_{j+2}, \ldots, v_{2k-2}\} \cap S) \cup \{v_{2k-1}\}$ through edges of color 'R'. It follows that $S \cup \{v_{2k-1}\}$ is a monochromatic clique of size at least k. □

Definition 1 ([18]). *Given events $E_1, \ldots, E_n \subset \Omega$ (where Ω is a finite probability space) and a subset $J \subset [n]$, the event E_i is said to be mutually independent of $\{E_j : j \in J\}$ if for all choices of disjoint subsets $J_1, J_2 \subset J$,*

$$P\left(E_i \cap \bigcap_{j_1 \in J_1} E_{j_1} \cap \bigcap_{j_2 \in J_2} \overline{E_{j_2}}\right) = P[E_i] \cdot P\left(\bigcap_{j_1 \in J_1} E_{j_1} \cap \bigcap_{j_2 \in J_2} \overline{E_{j_2}}\right).$$

Equipped with this notion, we state a commonly used form of the Lovász Local Lemma.

Theorem 4 (Symmetric version of Lovász Local Lemma [18]). *Suppose $p \in (0,1)$, $d \geq 1$, and $E_1, \ldots, E_n$ are events such that $P[E_i] \leq p$ for all i. If each E_i is mutually independent of all but d other events E_j, and $ep(d+1) \leq 1$, where $e = 2.71828\ldots$ is Euler's number, then*

$$P\left[\bigcap_{i=1}^{n} \overline{E_i}\right] > 0.$$

Theorem 5 ([18]). *If $e \cdot 2^{1-\binom{k}{2}} \cdot \binom{k}{2}\binom{n-2}{k-2} < 1$, then $R(k,k) > n$.*

Proof. Consider a random 2-edge coloring c of K_n with the vertex set $V(K_n) = \{1, 2, \ldots, n\}$ in which each edge of the K_n is colored with R or B uniformly at random. For distinct $i, j \in \{1, 2, \ldots, n\}$, define a random variable X_{ij} that takes the value $c(ij)$, i.e. the color of the edge ij. Clearly, $P[X_{ij} = R] = P[X_{ij} = B] = \frac{1}{2} \quad \forall ij \in E(K_n)$. For $S \subseteq V(K_n)$ such that $|S| = k$, let A_S be the event that the clique formed by the vertices of S is monochromatic in the coloring c.

Notice that the set $\mathcal{E} = \{A_S \mid S \subseteq V(G) \text{ and } |S| = k\}$ is a collection of $\binom{n}{k}$ events. Observe that each event $A_S \in \mathcal{E}$ is uniquely determined by the $\binom{k}{2}$ random variables in $\{X_{ij} \mid i, j \in S\}$. It can be seen that if $A_S \in \mathcal{E}$ and $\mathcal{A} \subseteq \mathcal{E}$, then A_S is mutually independent of the collection $\mathcal{A}$ of events if and only if for each $A_{S'} \in \mathcal{A}$, we have $|S \cap S'| < 2$.

We define the dependency graph G for $\mathcal{E}$ to be the graph on vertex set $\mathcal{E}$ with an edge between two events A_S and $A_{S'}$ in $\mathcal{E}$ if $S \neq S'$ but $|S \cap S'| \geq 2$. Define $\Gamma(A_S)$ to be the neighborhood of A_S in this dependency graph. Notice that $|\Gamma(A_S)| \leq \binom{k}{2}\left(\binom{n-2}{k-2} - 1\right) \leq \binom{k}{2}\binom{n-2}{k-2} - 1$. Notice that an $A_S \in \mathcal{E}$ is mutually independent of the events $\mathcal{E} \setminus \{\Gamma(A_S) \cup A_S\}$. Thus each event $A_S \in \mathcal{E}$ is mutually independent of all but $|\Gamma(A_S)|$ other events in $\mathcal{E}$. We also know that for each $A_S \in \mathcal{E}$, just like in the proof of the previous lemma, we have

$$P[A_S] = 2^{1-\binom{k}{2}}.$$

By the assumption in the statement of the lemma, we have

$$e \cdot P[A_S] \cdot (|\Gamma(A_S) + 1|) < 1.$$

Therefore, applying Theorem 4,

$$P\left[\bigcap_{A_S \in \mathcal{E}} \overline{A_S}\right] > 0.$$

Thus, there is a non-zero probability that none of the events A_S occurs; i.e. no clique on k vertices is monochromatic. Therefore, we can conclude that $R(k,k) > n$. □

Using the entropy compression method

We shall study how the entropy compression method can be applied to derive lower bounds on $R(k,k)$ without using the probabilistic techniques employed in the proofs of Theorems 2

and 5. As in the previous chapter, we shall construct an algorithm that takes as input a complete graph G of n vertices (*i.e.* K_n) and a string s, and attempts to construct a 2-edge coloring of G that does not contain a monochromatic clique on k vertices. In this case, the string $s \in \{R, B\}^*$. The algorithm chooses an arbitrary ordering $\prec$ on the edges of G. In the i-th iteration of the main loop of the algorithm, it chooses the uncolored edge that is smallest with respect to the ordering $\prec$ and assigns it the color $s[i]$. Our aim is to show that if n is smaller than some particular value, then the graph G gets colored without any monochromatic clique of k vertices for some input string s.

The algorithm and its analysis

Let s be a string of length t from the alphabet $\{R, B\}$. The algorithm orders the cliques of size k in G as $C_1, C_2, \ldots, C_{\binom{k}{2}}$ arbitrarily. At step i of the algorithm, the i-th entry $s[i]$ of s will be utilized to assign a color to the smallest edge with respect to $\prec$ among the uncolored edges of G. If a monochromatic clique of size k is created after the coloring of this edge, say C_j, then uncolor all the edges of C_j except the edge in C_j that is smallest with respect to $\prec$, and output j. The algorithm applied on G returns output $(j_1, j_2, j_2, \ldots, j_r, \Phi)$ i.e. the indices of the monochromatic cliques of size k formed, and the partial edge coloring Φ of G created by the algorithm at the time of its termination (here, r is the number of monochromatic cliques that were formed during the execution of the algorithm).

Algorithm 2 : Edge Coloring Algorithm for Complete Graph

procedure ECC(G, s) ▷ Input: Complete graph G and a string $s \in \{R, B\}^*$
  Choose an arbitrary ordering $\prec$ on the edges of G
  Order the cliques of size k of G as $C_1, C_2, \ldots, C_{\binom{n}{k}}$
  Initialize the color of every edge to 'U' ▷ U stands for "uncolored"
  $i \leftarrow 0$
  while G is not fully colored **and** $i < t$ **do**
    $i \leftarrow i + 1$
    Let e_i be the first uncolored edge in the edge ordering $\prec$
    Color e_i with the color $s[i]$
    if a monochromatic clique C (containing e_i) of size k is present in the graph **then**
      Get the smallest edge $e^{\min}$ in $E(C) \setminus \{e_i\}$ with respect to $\prec$
      Uncolor all edges in C except $e^{\min}$
      Output the index of C in the ordering $C_1, C_2, \ldots, C_{\binom{n}{k}}$
    end if
  end while
  Output for each edge e in G the color assigned to it; this is a string in $\{R, B, U\}^{\binom{k}{2}}$.
end procedure

Suppose that for each $q \in \{1, 2, \ldots, r\}$, the q-th instance of the appearance of a monochromatic clique appears in the i_q-th iteration of the main loop of the algorithm (the monochromatic clique that appears during this iteration is C_{j_q}). We let $i_0 = 0$. For each $q \in \{1, 2, \ldots, r\}$, let X_q be the set of uncolored edges after the i_q-th iteration of the main loop. We define

$X_0 = E(G)$. Also, let Φ_q be the partial edge coloring of G after the i_q-th iteration of the main loop. We let Φ_0 denote the coloring in which every edge is uncolored. For each $q \in \{1, 2, \ldots, r\}$, we denote $e_q^{\min} = \min_{\prec}\{E(C_q)\} \setminus \{e_{i_q}\}$ and $e_q^{\max} = \max_{\prec}\{E(C_{j_q})\}$. For ease of writing, for any edge $e \in E(G)$, we define $E(G)_{\preceq e}$ to be the edges in G that are smaller than or equal to e with respect to the ordering $\prec$. Let $f(s)$ denote the output string of the algorithm when given s as the input string. We will prove that if the algorithm does not fully color the graph G for some input string s, then the output string $f(s) = (j_1, j_2, j_2, \ldots, j_r, \Phi)$ uniquely determines s. Note that given $f(s)$, we can determine $j_1, j_2, \ldots, \Phi$, since Φ consists of the last $\binom{n}{2}$ symbols on the string, and the symbols before that on the string are $j_1, j_2, \ldots, j_r$.

Suppose that the graph G is not fully colored by the algorithm when given a string s as input. Let $t = |s|$. Then it is clear that the main loop of the algorithm is executed t times.

Lemma 6. $r \leq \frac{t}{\binom{k}{2}-1}$.

Proof. For each $q \in \{1, 2, \ldots, r\}$, we define for each edge $e \in E(C_{j_q}) \setminus \{e_q^{\min}\}$, the index $f(e, q) = \max\{i \in \{1, 2, \ldots, i_q\} \mid e_i = e\}$. Further, define $F(q) = \{f(e, q) \mid e \in E(C_{j_q}) \setminus \{e_q^{\min}\}\}$. It is clear that $|F(q)| = \binom{k}{2} - 1$. We claim that for distinct $q, q' \in \{1, 2, \ldots, r\}$, $F(q) \cap F(q') = \emptyset$. Suppose for the sake of contradiction that $i \in F(q) \cap F(q')$. Then clearly, $e_i \in E(C_{j_q}) \cap E(C_{j_{q'}})$. Let us assume without loss of generality that $q < q'$. Since $i \in F(q) \cap F(q')$, we have that $f(e_i, q) = f(e_i, q') = i$, and therefore $i \leq i_q$. If there exists $i' \in \{i_q + 1, i_q + 2, \ldots, i_{q'}\}$ such that $e_{i'} = e_i$, then it must be the case that $i = f(e_i, q') = f(e_{i'}, q') \geq i' > i_q$, which contradicts our earlier observation that $i \leq i_q$. So there does not exist $i' \in \{i_q + 1, i_q + 2, \ldots, i_{q'}\}$ such that $e_{i'} = e_i$. Since the edge e_i is uncolored after the i_q-th iteration (notice that $e_i \neq e_q^{\min}$), this means that the edge e_i is uncolored at the time that the monochromatic clique $C_{j_{q'}}$ is about to be uncolored in the $i_{q'}$-th iteration of the main loop. As $e_i \in E(C_{j_{q'}})$, this contradicts the fact that all edges of the clique $C_{j_{q'}}$ are colored when it is just about to be uncolored. This shows that for distinct $q, q' \in \{1, 2, \ldots, r\}$, $F(q) \cap F(q') = \emptyset$. As $F(q) \subseteq \{1, 2, \ldots, t\}$ for each $q \in \{1, 2, \ldots, r\}$, it now follows that $r \leq \frac{t}{\binom{k}{2}-1}$. □

Lemma 7. *For each $q \in \{0, 1, \ldots, r\}$, the set X_q is uniquely determined by the output string $f(s)$.*

Proof. To prove this result, we apply induction on q. As the base case, we have that $X_0 = E(G)$ (and so it is uniquely determined, no matter what $f(s)$ is). Let $q \in \{1, 2, \ldots, r\}$. We assume inductively that X_{q-1} is uniquely determined given $f(s)$. We shall show that X_q is uniquely determined given X_{q-1} and $f(s)$. It is easy to see that $\max\{X_{q-1} \cap E(C_{j_q})\} = e_{i_q}$. Clearly, when the clique C_{j_q} is about to be uncolored, every edge in $E(G)_{\preceq e_{i_q}}$ must be colored. Hence $X_q = \left(X_{q-1} \setminus E(G)_{\preceq e_{i_q}}\right) \cup \left(E(C_{j_q}) \setminus e_q^{\min}\right)$. This shows that X_q is uniquely determined for each $q \in \{0, 1, \ldots, r\}$ given $f(s)$. □

Recall that Φ is the partial edge coloring of G after the t-th iteration of the main loop. We define $\Phi_{r+1} = \Phi$. Similarly, let X_{r+1} denote the set of uncolored edges after the t-th iteration

of the main loop. Note that X_{r+1} is uniquely determined given $f(s)$, since it can be directly derived from the partial edge coloring Φ of G that is part of $f(s)$.

Lemma 8. *For each $q \in \{1, 2, \ldots, r\}$, the partial edge coloring Φ_q is uniquely determined by the output string $f(s)$.*

Proof. Given $f(s)$, we have the partial edge coloring Φ_{r+1} of G. It can be seen that we can uncolor all the edges in the set $X_r \setminus X_{r+1}$ from the coloring Φ_{r+1} to obtain the coloring Φ_r. Now suppose that Φ_{q+1} is uniquely determined given $f(s)$ for some $q \in \{1, 2, 3, \ldots, r-1\}$. By Lemma 7, we know that $X_1, X_2, \ldots, X_q$ are uniquely determined given $f(s)$. We shall be done if we show that Φ_q is uniquely determined given Φ_{q+1}, X_{q+1} and X_q. It is easy to see that Φ_q can be obtained from Φ_{q+1} by uncoloring all the edges of the set $(X_q \setminus X_{q+1})$ and then coloring all the edges of $C_{j_{q+1}} \setminus X_q$ using the color of the edge $e^{\min}_{q+1}$ in Φ_{q+1}. This completes the proof. □

Lemma 9. *The input string s of the algorithm is uniquely determined by the output string $f(s)$.*

Proof. For each $q \in \{1, 2, \ldots, r\}$, we define Φ^*_q to be the partial edge coloring of G constructed by the algorithm just before it uncolors the edges of the clique C_{j_q}. It can be seen that for each $q \in \{1, 2, \ldots, r\}$, Φ^*_q can be obtained from Φ_q by coloring the edges in $E(C_{j_q}) \setminus \{e^{\min}_q\}$ using the color of the edge $e^{\min}_q$ in Φ_q. For each $q \in \{0, 1, \ldots, r\}$, the edges of G that get colored during iterations $i_q + 1$ to i_{q+1} are exactly the ones in $Y = (X_q \setminus X_{q+1}) \cup (E(C_{j_{q+1}}) \cap X_q)$ (these are exactly the edges that are colored in Φ^*_{q+1} but not in Φ_q). Hence we have that $|Y| = i_{q+1} - i_q$. Let $Y = \{e^1, e^2, \ldots, e^{|Y|}\}$ where $e^1 \prec e^2 \prec \cdots \prec e^{|Y|}$. It is clear from the algorithm that the color of the edge e^p in Φ^*_{q+1}, where $p \in \{1, 2, \ldots, |Y|\}$, is $s[i_q + p]$. Since we know by Lemma 7 and Lemma 8 that for each $q \in \{0, 1, \ldots, r+1\}$, X_q and Φ_q are determined uniquely given $f(s)$, it follows that Φ^*_q is also determined uniquely for each $q \in \{1, 2, \ldots, r\}$, given $f(s)$. Now we have from the argument above that $s[i_q + 1], s[i_q + 2], \ldots, s[i_{q+1}]$ are determined uniquely given $f(s)$, for each $q \in \{0, 1, \ldots, r-1\}$. This means that $s[1], s[2], \ldots, s[i_r]$ are determined uniquely given $f(s)$. The edges that get colored during the iterations $i_r + 1$ to t of the algorithm are exactly the edges in $Y = X_r \setminus X_{r+1}$. Here, $|Y| = t - i_r$. Again if $Y = \{e^1, e^2, \ldots, e^{|Y|}\}$ where $e^1 \prec e^2 \prec \cdots \prec e^{|Y|}$, we get that the color of e^p in Φ is exactly $s[i_r + p]$, for each $p \in \{1, 2, \ldots, |Y|\}$. This proves that s is uniquely determined given $f(s)$. □

Corollary 2. *Let s_1 and s_2 be two input strings of length t. If $s_1 \neq s_2$, then $f(s_1) \neq f(s_2)$.*

Let Z be the total number of possible input strings of length t, so $|Z| = 2^t$. And also suppose that $f(Z)$ be the total number of output strings corresponding to input strings in Z. Recall each output string is of the form $(j_1, j_2, \ldots, j_r, \Phi)$, where each j_q has $\binom{n}{k}$ possible values and Φ can be one among $3^{\binom{n}{2}}$ possibilities (because $|E(G)| = \binom{n}{2}$ and the edges are of 3 types, i.e. R or B or uncolored). Hence using Lemma 6, we have $|f(Z)| \leq 3^{\binom{n}{2}} \cdot \binom{n}{k}^{\frac{t}{\binom{k}{2}-1}}$.

Recall that we have assumed that the algorithm does not fully color the graph G for any string in Z. From Corollary 2, we know that $|f(Z)| \geq |Z|$. Thus:

$$\begin{aligned} 3^{\binom{n}{2}} \binom{n}{k}^{\frac{t}{\binom{k}{2}-1}} &\geq 2^t \\ \Rightarrow 3^{\frac{\binom{n}{2}\left(\binom{k}{2}-1\right)}{t}} \binom{n}{k} 2^{1-\binom{k}{2}} &\geq 1 \\ \Rightarrow \binom{n}{k} 2^{1-\binom{k}{2}} &\geq \left(\frac{1}{3}\right)^{\frac{\binom{n}{2}\left(\binom{k}{2}-1\right)}{t}} \end{aligned}$$

Now suppose that $\binom{n}{k}2^{1-\binom{k}{2}} < 1$. Let $\epsilon = 1 - \binom{n}{k}2^{1-\binom{k}{2}}$. Then clearly, $\binom{n}{k}2^{1-\binom{k}{2}} = 1 - \epsilon$ and $\epsilon > 0$. Then from the above inequality, we have

$$\begin{aligned} \left(\frac{1}{3}\right)^{\frac{\binom{n}{2}\left(\binom{k}{2}-1\right)}{t}} &\leq 1 - \epsilon \\ \Rightarrow \left(\frac{1}{3}\right)^{\binom{n}{2}\left(\binom{k}{2}-1\right)} &\leq (1-\epsilon)^t \\ \Rightarrow \binom{n}{2}\left(\binom{k}{2}-1\right)\log\frac{1}{3} &\leq t\log(1-\epsilon) \\ \Rightarrow \binom{n}{2}\left(\binom{k}{2}-1\right)\log(3) &\geq t\log\left(\frac{1}{1-\epsilon}\right) \\ \Rightarrow \binom{n}{2}\left(\binom{k}{2}-1\right)\frac{\log(3)}{\log\left(\frac{1}{1-\epsilon}\right)} &\geq t \end{aligned}$$

This means if $t > \binom{n}{2}\left(\binom{k}{2}-1\right)\frac{\log(3)}{\log\left(\frac{1}{1-\epsilon}\right)}$, then it is not possible that the algorithm fails to fully color the graph G for every input string of length t. From this, we can conclude that if $\binom{n}{k}2^{1-\binom{k}{2}} < 1$, then there is some input string for which the algorithm fully colors the graph G; i.e. or in other words, the complete graph on n vertices has a 2-edge coloring in which there is no monochromatic clique of size k. We thus have an alternative proof for Theorem 2.

So the inequality becomes

$$\begin{aligned} \binom{n}{k}2^{1-\binom{k}{2}} &\geq 1 \\ \Rightarrow (\frac{en}{k})^k \cdot 2^{1-\binom{k}{2}} &\geq 1((\text{using Stirling's approximation})) \\ \Rightarrow n &\geq \frac{k}{e} \cdot 2^{\frac{(k+1)(k-2)}{2k}} \\ \Rightarrow n &> \frac{k}{2e} \cdot 2^{\frac{k}{2}} \end{aligned}$$

A modified algorithm

We now modify the previous algorithm *ECC(G, s)* a little bit and design a new algorithm *MECC(G, s)*. We output a "0" every time we color an edge in *MECC(G, s)* and when a monochromatic clique is detected we output the index of the clique as before. But unlike in the algorithm *ECC(G, s)*, where the index is from a global list of cliques, here the index is from a list of cliques that the current edge is a part of. We analyse the algorithm *MECC(G, s)* similarly as we did in previous algorithm to get a better lower bound on $R(k, k)$.

Algorithm 3 : Modified Edge Coloring Algorithm for Complete Graph

procedure MECC(G, s) ▷ Input: Complete graph G and a string $s \in \{R, B\}^*$
  Choose an arbitrary ordering $\prec$ on the edges of G
  For each edge $e \in E(G)$, choose an arbitrary ordering $C_1^e, C_2^e, \ldots, C_{\binom{n-2}{k-2}}^e$ of the cliques of size k of G that contain e
  Initialize the color of every edge to 'U' ▷ U stands for "uncolored"
  $i \leftarrow 0$
  while G is not fully colored **and** $i < t$ **do**
    $i \leftarrow i + 1$
    Let e_i be the first uncolored edge in the edge ordering $\prec$
    Color e_i with the color $s[i]$
    Output "0"
    if a monochromatic clique C (containing e_i) of size k is present in the graph **then**
      Get the smallest edge $e^{\min}$ in $E(C) \setminus \{e_i\}$ with respect to $\prec$
      Uncolor all edges in C except $e^{\min}$
      Output the index of C in the ordering $C_1^{e_i}, C_2^{e_i}, \ldots, C_{\binom{n-2}{k-2}}^{e_i}$
    end if
  end while
  Output for each edge e in G the color assigned to it; this is a string in $\{R, B, U\}^{\binom{k}{2}}$.
end procedure

Suppose that the MECC(G, s) algorithm when applied to a complete graph G on n vertices, and input string s, fails to fully color the graph. Let $g(s)$ be the string that is printed by the algorithm, and let $|s| = t$. The main loop of the algorithm executes exactly t times, and therefore the algorithm prints exactly t zeroes before the final step, in which it outputs the partial edge coloring. Suppose that the algorithm detects the formation of a monochromatic clique of size k a total of r times. If we denote the partial edge coloring of G created by the algorithm at the time of its termination as Φ, then the output string is of the form

$$000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0\Phi$$

where each j_q represents the indices of the monochromatic cliques formed and each block of consecutive zeroes has length at least 1, except the last block of zeroes, which may not necessarily exist.

Analysis

Suppose that for each $q \in \{1, 2, \ldots, r\}$, the q-th instance of the appearance of a monochromatic clique appears in the i_q-th iteration of the main loop of the algorithm (the monochromatic clique that appears during this iteration is $C_{j_q}^{e_{i_q}}$. We let $i_0 = 0$. We define X_i to be the set of uncolored edges after the i-th iteration of the main loop of the algorithm and also define Φ_i to be the partial edge coloring of G after this iteration. Note that $\Phi = \Phi_t$. We let $X_0 = E(G)$. We will prove that output string $g(s) = 000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0\Phi$ uniquely determines the corresponding input string s used in the algorithm. For each $q \in \{1, 2, \ldots, r\}$, we let $e^{\min}(C_{j_q}^{e_{i_q}}) = \min_{\prec}\left(E(C_{j_q}^{e_{i_q}}) \setminus \{e_{i_q}\}\right)$.

Note that Φ can be thought of as a string in $\{U, R, B\}^{\binom{n}{2}}$. Hence Φ is one of $3^{\binom{n}{2}}$ possible strings of length $\binom{n}{2}$. Also note that for each $q \in \{1, 2, \ldots, r\}$, we have $1 \leq j_q \leq \binom{n-2}{k-2}$.

Blocks of $g(s)$:

Let $block(i)$ be the substring of $g(s)$ that is printed by the algorithm in the i-th iteration of its main loop. Thus $g(s)$ is of the form $block(1)block(2)\ldots block(t)\Phi$. One can think of $block(i)$ as a record of the actions performed by the algorithm in the i-th iteration of its main loop. Note that given $g(s)$, the strings $block(1), block(2), \ldots, block(t)$ can be determined easily: every iteration that prints non-zero symbols prints a 0 followed by an index j_q, and every other iteration prints a single 0. Thus, for each $i \in \{1, 2, \ldots, t\}$, the string $block(i)$ is either 0 or an index prefixed with a 0, and these can be derived easily from the string $g(s)$.

Lemma 10. *For each $i \in \{1, 2, \ldots, t\}$, X_i is uniquely determined by the output $g(s)$.*

Proof. Clearly, $X_0 = E(G)$. We claim that given X_{i-1} and $g(s)$, X_i is uniquely determined for each $i \in \{1, 2, 3, \ldots, t\}$. Note that this will complete the proof of the lemma. Notice that in the i-th iteration of the main loop, the algorithm chooses as e_i the smallest edge in the ordering $\prec$ that is uncolored. Since X_{i-1} is the set of uncolored edges at the end of the $(i-1)$-th iteration of the loop, and also at the beginning of the i-th iteration of the loop, it is clear that $e_i = \min_{\prec} X_{i-1}$. Thus, e_i is uniquely determined given $g(s)$ and X_{i-1}. If $block(i)$ is of the form "0", then $X_i = X_{i-1} \setminus \{e_i\}$, since e_i is the only edge that gets colored in the i-the iteration of the loop and no edges are uncolored (as otherwise $block(i)$ would not have been just 0). Since e_i is uniquely determined given $g(s)$ and X_{i-1}, we have that X_i is uniquely determined given $g(s)$ and X_{i-1}, and we are done in this case. Otherwise, $block(i) = 0j$, where $j \in \{j_1, j_2, \ldots, j_r\}$. This means that a monochromatic clique of size k, namely $C_j^{e_i}$ was detected after the edge e_i was colored. Then $X_i = X_{i-1} \cup (E(C_j^{e_i}) \setminus e^{\min}(C_j^{e_i}))$, since every edge of C_j other than $e^{\min}(C_j^{e_i})$ get uncolored during this iteration of the algorithm, and no other edges are uncolored. This means that X_i is uniquely determined given $g(s)$ and X_{i-1}. □

Lemma 11. *For each $i \in \{1, 2, \ldots, t\}$, Φ_{i-1} and $s[i]$ are uniquely determined given $g(s)$.*

Proof. Using the previous lemma we know that $e_1, e_2, \ldots, e_t$ are all uniquely determined given $g(s)$. Clearly, $\Phi_t = \Phi$, and therefore Φ_t is uniquely determined given $g(s)$. For convenience, we also assume that $s[t+1]$ (which does not exist) is also uniquely determined given $g(s)$. We assume inductively that Φ_i and $s[i+1]$ are uniquely determined given $g(s)$ and prove that then Φ_{i-1} and $s[i]$ are also uniquely determined. Now if $block(i) = 0$, that means e_i is the only edge that gets colored in the i-th iteration of the loop and no edges are uncolored (as otherwise $block(i)$ would not have been just 0), then one can obtain Φ_{i-1} from Φ_i by just uncoloring the edge e_i. Moreover, $s[i]$ is the color of the edge e_i in Φ_i. Otherwise, $block(i) = 0j$, where $j \in \{j_1, j_2, \ldots, j_r\}$. This means that a monochromatic clique of size k, namely $C_j^{e_i}$, was detected after the edge e_i was colored in i-th iteration. Since every edge of $C_j^{e_i}$ other than $e^{\min}(C_j^{e_i})$ gets uncolored during this iteration of the algorithm, one can obtain Φ_{i-1} from Φ_i by coloring every edge in $E(C_j^{e_i}) \setminus \{e^{\min}(C_j^{e_i})\}$ with the color of the edge $e^{\min}(C_j^{e_i})$ in Φ_i. Again, $s[i]$ is the color of the edge $e^{\min}(C_j^{e_i})$ in Φ_i. This completes the proof. □

Corollary 3. *Let s_1 and s_2 be two input strings of length t. If $s_1 \neq s_2$, then $g(s_1) \neq g(s_2)$.*

It is not hard to see that using arguments similar to that in the proof of Lemma 6, we can obtain the same result; so we give the statement without proof.

Lemma 12. $r \leq \frac{t}{\binom{k}{2}-1}$.

Let Z be the total number of possible input strings of length t, so $|Z| = 2^t$. And also suppose that $g(Z)$ be the total number of output strings corresponding to input strings in Z. Recall that each output string is of the form $(000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0\Phi)$, where each j_q has $\binom{n-2}{k-2}$ possible values. As observed before, Φ has $3^{\binom{n}{2}}$ possible values. Consider $g^*(s)$ be the string excluding Φ from $g(s)$, and then replacing every non-zero symbol by '1', as follows:

$$g(s) = 000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0 \underbrace{\Phi}_{\text{is removed}}$$
$$g^*(s) = 000\ldots0\,1\,000\ldots0\,1\,0\ldots1\,000\ldots0$$

Notice that two j_q's cannot be printed consecutively and $block(1)$ is always 0. And recall that the algorithm prints exactly t 0's after the termination of the algorithm. So there are at most $\binom{t}{r}$ many total possible values for $g^*(Z)$. Then,

$$|g(Z)| \leq 3^{\binom{n}{2}} \binom{t}{r} \binom{n-2}{k-2}^r$$

Therefore, using Lemma 12 (observe that $\frac{t}{\binom{k}{2}-1} > 2$),

$$|g(Z)| \leq 3^{\binom{n}{2}} \binom{t}{\frac{t}{\binom{k}{2}-1}} \binom{n-2}{k-2}^{\frac{t}{\binom{k}{2}-1}}$$

Recall that we have assumed that the algorithm does not fully color the graph G for any string in Z. From Corollary 2, we know that $|f(Z)| \geq |Z|$. Thus:

$$\begin{aligned}
3^{\binom{n}{2}} \binom{t}{\frac{t}{\binom{k}{2}-1}} \binom{n-2}{k-2}^{\frac{t}{\binom{k}{2}-1}} &\geq 2^t \\
3^{\binom{n}{2}} \left(\frac{et}{\frac{t}{\binom{k}{2}-1}}\right)^{\frac{t}{\binom{k}{2}-1}} \binom{n-2}{k-2}^{\frac{t}{\binom{k}{2}-1}} &\geq 2^t \text{(using Stirling's approximation)} \\
\Rightarrow 3^{\frac{\binom{n}{2}\left(\binom{k}{2}-1\right)}{t}} \cdot e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}} &\geq 1 \\
\Rightarrow e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}} &\geq \left(\frac{1}{3}\right)^{\frac{\binom{n}{2}\left(\binom{k}{2}-1\right)}{t}}
\end{aligned}$$

Now suppose that $e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}} < 1$. Let $\epsilon = 1 - e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}}$. Then clearly, $e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}} = 1 - \epsilon$ and $\epsilon > 0$. Then from the above inequality, we have

$$\begin{aligned}
\left(\frac{1}{3}\right)^{\frac{\binom{n}{2}\left(\binom{k}{2}-1\right)}{t}} &\leq 1 - \epsilon \\
\Rightarrow \left(\frac{1}{3}\right)^{\binom{n}{2}\left(\binom{k}{2}-1\right)} &\leq (1-\epsilon)^t \\
\Rightarrow \binom{n}{2}\left(\binom{k}{2}-1\right)\log\frac{1}{3} &\leq t\log(1-\epsilon) \\
\Rightarrow \binom{n}{2}\left(\binom{k}{2}-1\right)\log(3) &\geq t\log\left(\frac{1}{1-\epsilon}\right) \\
\Rightarrow \binom{n}{2}\left(\binom{k}{2}-1\right)\frac{\log(3)}{\log\left(\frac{1}{1-\epsilon}\right)} &\geq t
\end{aligned}$$

This means if $t > \binom{n}{2}\left(\binom{k}{2}-1\right)\frac{\log(3)}{\log\left(\frac{1}{1-\epsilon}\right)}$, then it is not possible that the algorithm fails to fully color the graph G for every input string of length t. From this, we can conclude that if $e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}} < 1$, then there is some input string for which the algorithm fully colors the graph G; or in other words, the complete graph on n vertices has a 2-edge coloring in which there is no monochromatic clique of size k. We thus have an alternative proof for Theorem 5.

So the inequality becomes

$$\begin{aligned}
e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}} &\geq 1 \\
\Rightarrow e \cdot \left(\binom{k}{2}-1\right) \cdot \left(\frac{e(n-2)}{k-2}\right)^{k-2} \cdot 2^{1-\binom{k}{2}} &\geq 1 \text{(using Stirling's approximation)}
\end{aligned}$$

$$\begin{aligned} \Rightarrow n &\geq 2 + \frac{k-2}{e}\left(\frac{2^{\binom{k}{2}-1}}{e(\binom{k}{2}-1)}\right)^{\frac{1}{k-2}} \\ \Rightarrow n &\geq 2 + o(1)\cdot(k-2)\cdot 2^{\frac{k+1}{2}} \end{aligned}$$

By the LLL, we had obtained a bound of $e \cdot 2^{1-\binom{k}{2}} \cdot \binom{k}{2}\binom{n-2}{k-2} < 1$ (Theorem 5), and we are also getting similar bounds, i.e. $e \cdot \left(\binom{k}{2}-1\right) \cdot \binom{n-2}{k-2} \cdot 2^{1-\binom{k}{2}} < 1$. Our future plan is to count $g(Z)$ accurately using *N-Dyck words* (see the conclusion) to enhance the bound.

*****

k-Uniform Hypergraph

In the domain of combinatorial mathematics, hypergraphs provide a framework for studying relationships between objects where hyperedges (or hyperedges) can connect more than two hyperedges. A *k-uniform hypergraph* $G = (V(G), E(G))$ is a mathematical structure consisting of a finite set of vertices, $V(G)$, and a collection of k-element subsets of $V(G)$ known as hyperedges, denoted by $E(G)$.

This work investigates the concept of 2-coloring for k-uniform hypergraphs. A 2-coloring of G assigns a color, either R or B, to each vertex in $V(G)$. A valid 2-coloring satisfies the following property: no hyperedge in $E(G)$ is monochromatic, meaning every hyperedge must contain vertices colored both red and blue. A k-uniform hypergraph that admits such a 2-coloring is classified as 2-colorable.

Let G be a k-uniform hypergraph with vertex set V and edge set E. We consider the scenario where each vertex of G is colored uniformly at random, either R or B. Suppose E_i denotes the event that the hyperedge $H_i \in E(G)$ is monochromatic, i.e., all vertices incident to this hyperedge have the same color. Since each vertex is independently colored with equal probability of being R or B, the probability that the hyperedge H_i is monochromatic is 2^{1-k}.

Theorem 6. *If G has fewer than $2^k - 1$ hyperedges, then in a 2-coloring of G, there exists at least one monochromatic hyperedge in G with probability less than 1.*

Proof. Since G has fewer than $2^k - 1$ hyperedges, we can employ the union bound to estimate the probability that there exists at least one monochromatic hyperedge in G. By the union bound, the probability that the union of all events E occurs is bounded above by the sum of the individual probabilities:

$$\sum_{i=1}^{|E|} P(E_i) < |E| \cdot 2^{1-k}$$

where $|E|$ denotes the number of hyperedges in G. Substituting $|E| < 2^k - 1$, we get:

$$|E| \cdot 2^{1-k} < (2^k - 1) \cdot 2^{1-k} = 1$$

Thus, the probability that there exists at least one monochromatic hyperedge in G is less than 1. Consequently, we can conclude that there exists a valid 2-coloring of G i.e. G is 2-colorable. □

However, this argument breaks down when G has more than $2^k - 1$ hyperedges, as the probability bound obtained from the union bound may exceed 1, rendering the conclusion of G being a valid 2-colorable invalid. But using the symmetric version of the Lovász Local Lemma (refer to Theorem 4), the following result is found.

Theorem 7. *Suppose G be a k-uniform hypergraph with maximum degree d (every vertex of G has maximum degree d) $|E(G)| = m$ and $\forall i \leq m$ hyperedge $H_i \in E(G)$, E_i denotes the event that H_i is monochromatic. And $\forall i \leq m$ E_i is mutually independent of all other events except D of them, where $D + 1 < \frac{2^{k-1}}{e}$, then G is 2-colorable.*

Proof. Given total m events, which are $E_1, E_2, \ldots, E_m$, $\forall i \leq m$ E_i is mutually independent of all other events except d of them. Consider fixing an hyperedge H_i; now, any vertex-coloring on the hyperedges disjoint from H_i is independent of E_i, since the node colors are independent and identically distributed (i.i.d.) Bernoulli random variables. We also have that $P[E_i] = 2^{1-k}$. Therefore, the assumptions of the Symmetric Lovász Local Lemma are satisfied as long as:

$$D + 1 < \frac{2^{k-1}}{e} \Rightarrow e \cdot (D + 1) \cdot P[E_i] < 1$$

Hence

$$P\left[\bigcap_{i=1}^{m} \overline{E_i}\right] > 0.$$

That is the probability that none of these events $E_1, E_2, \ldots, E_m$ happens is non-zero. Hence, we conclude that G is 2-colorable. □

It is important to note that this condition is invariant with respect to the number of hyperedges in the hypergraph G.

We shall study how the entropy compression method can be applied to derive upper bounds on d without using the probabilistic techniques employed in the proof of Theorem 7. As in the previous chapter, we shall construct an algorithm that takes as input a k-uniform hypergraph G with $|V(G)| = n$, $|E(G)| = m$ and maximum degree d and a string s, and attempts to construct a 2-coloring of G that does not contain any monochromatic hyperedge. In this case, the string $s \in \{R, B\}^*$. The algorithm chooses an arbitrary ordering $\prec$ on the vertices of G. In the i-th iteration of the main loop of the algorithm, it chooses the uncolored vertex that is smallest with respect to the ordering $\prec$ and assigns it the color $s[i]$. Our aim is to show that if d is smaller than some particular value, then the graph G gets colored without any monochromatic hyperedge for some input string s. Notice that a hyperedge H containing a vertex v, can intersect with at most $k(d-1)$ ($= D$, say) many other hyperedges.

Algorithm 4 : Vertex Coloring Algorithm for k-uniform hypergraph

procedure VHPG(G, s) ▷ Input: k-uniform hypergraph$G$ and a string $s \in \{R, B\}^*$
  Choose an arbitrary ordering $\prec$ on the vertices of G
  For each vertex $v \in V(G)$, choose an arbitrary ordering $H_1^v, H_2^v, \ldots, H_D^v$ of the hyperedges of G that contain v
  ▷ $D = k(d-1)$
  Choose an arbitrary ordering $\prec'$ on the hyperedges of G
  Initialize the color of every vertex to 'U' ▷ U stands for "uncolored"
  $i \leftarrow 0$
  while G is not fully colored **and** $i < t$ **do**
    $i \leftarrow i + 1$
    Let v_i be the first uncolored vertex in the vertex ordering $\prec$
    Color v_i with the color $s[i]$
    Output "0"
    if a monochromatic hyperedge H (containing v_i) of size k is present in the graph **then**
      Get the smallest vertex $v^{\min}$ in $V(H) \setminus \{v_i\}$ with respect to $\prec$
      Uncolor all vertices in H except $v^{\min}$
      Output the index of H in the ordering $H_1^v, H_2^v, \ldots, H_D^v$
    end if
  end while
  Output for each vertex v in G the color assigned to it; this is a string in $\{R, B, U\}^n$.
end procedure

Suppose that the VHPG(G, s) algorithm is applied to the k-uniform hypergraph G. Let $g(s)$ be the string that is printed by the algorithm, and let $|s| = t$. The main loop of the algorithm executes exactly t times, and therefore the algorithm prints exactly t zeroes before the final step, in which it outputs the partial vertex coloring. Suppose that the algorithm detects the formation of a monochromatic vertex a total of r times and the partial vertex coloring Φ of G created by the algorithm at the time of its termination, then the example output is as the following

$$000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0\Phi$$

where each j_q represents the indices of the monochromatic hyperedges formed and each block of consecutive zeroes has length at least 1, except the last block of zeroes, which may not necessarily exist.

Analysis

Suppose that for each $q \in \{1, 2, \ldots, r\}$, the q-th instance of the appearance of a monochromatic hyperedge appears in the i_q-th iteration of the main loop of the algorithm (the monochromatic hyperedge that appears during this iteration is $H_{j_q}^{e_{i_q}}$. We let $i_0 = 0$. We define X_i to be the set of uncolored vertices after the i-th iteration of the main loop of the algorithm and also define Φ_i to be the partial vertex coloring of G after this iteration. Note that $\Phi = \Phi_t$. We let $X_0 =$

$V(G)$. We will prove that output string $g(s) = 000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0\Phi$ uniquely determines the corresponding input string s used in the algorithm. For each $q \in \{1, 2, \ldots, r\}$, we let $v^{\min}(H_{j_q}^{v_{i_q}}) = \min_{\prec}\left(V(H_{j_q}^{v_{i_q}}) \setminus \{v_{i_q}\}\right)$.

The partial vertex coloring Φ of G that is output at the final step is a string of length n where $\Phi[i]$ is color of the i-th vertex in the vertex ordering $\prec$, where the symbol U is used to denote the fact that the corresponding vertex is uncolored. Thus $\Phi \in \{U, R, B\}^n$. Hence Φ is one of 3^n possible strings of length n.

Blocks of $g(s)$:

Let $block(i)$ be the substring of $g(s)$ that is printed by the algorithm in the i-th iteration of its main loop. Thus $g(s)$ is of the form $block(1)block(2)\ldots block(t)\Phi$. One can think of $block(i)$ as a record of the actions performed by the algorithm in the i-th iteration of its main loop. Note that given $g(s)$, the strings $block(1), block(2), \ldots, block(t)$ can be determined easily: every iteration that prints non-zero symbols prints a 0 followed by an index j_q, and every other iteration prints a single 0. Thus, for each $i \in \{1, 2, \ldots, t\}$, the string $block(i)$ is either 0 or an index prefixed with a 0, and these can be derived easily from the string $g(s)$.

Lemma 13. *For each $i \in \{1, 2, \ldots, t\}$, X_i is uniquely determined by the output $g(s)$.*

Proof. Clearly, $X_0 = E(G)$. We claim that given X_{i-1} and $g(s)$, X_i is uniquely determined for each $i \in \{1, 2, 3, \ldots, t\}$. Note that this will complete the proof of the lemma. Notice that in the i-th iteration of the main loop, the algorithm chooses as v_i the smallest vertex in the ordering $\prec$ that is uncolored. Since X_{i-1} is the set of uncolored vertices at the end of the $(i-1)$-th iteration of the loop, and also at the beginning of the i-th iteration of the loop, it is clear that $v_i = \min_{\prec} X_{i-1}$. Thus, v_i is uniquely determined given $g(s)$ and X_{i-1}. If $block(i)$ is of the form "0", then $X_i = X_{i-1} \setminus \{v_i\}$, since v_i is the only vertex that gets colored in the i-the iteration of the loop and no vertices are uncolored (as otherwise $block(i)$ would not have been just 0). Since v_i is uniquely determined given $g(s)$ and X_{i-1}, we have that X_i is uniquely determined given $g(s)$ and X_{i-1}, and we are done in this case. Otherwise, $block(i) = 0j$, where where $j \in \{j_1, j_2, \ldots, j_r\}$. This means that a monochromatic hyperedge, $H_j^{v_i}$ was detected after the vertex v_i was colored. Then $X_i = X_{i-1} \cup (V(H_{j_q}^{v_i}) \setminus v^{\min}(H_j^{v_i})))$, since every vertex of $H_j^{v_i}$ other than $v^{\min}(H_j^{v_{i_q}})$ get uncolored during this iteration of the algorithm, and no other vertices are uncolored. This means that X_i is uniquely determined given $g(s)$ and X_{i-1}. □

Lemma 14. *For each $i \in \{1, 2, \ldots, t\}$, Φ_{i-1} and $s[i]$ are uniquely determined given $g(s)$.*

Proof. Using the previous lemma we know that $v_1, v_2, \ldots, v_t$ are all uniquely determined given $g(s)$. Clearly, $\Phi_t = \Phi$, and therefore Φ_t is uniquely determined given $g(s)$. For convenience, we also assume that $s[t+1]$ (which does not exist) is also uniquely determined given $g(s)$. We assume inductively that Φ_i and $s[i+1]$ are uniquely determined given $g(s)$ and prove that then Φ_{i-1} and $s[i]$ are also uniquely determined. Now if $block(i) = 0$, that means v_i is the only vertex that gets colored in the i-th iteration of the loop and no vertices are uncolored (as otherwise $block(i)$ would not have been just 0), then one can obtain Φ_{i-1} from Φ_i by

just uncoloring the vertex v_i. Moreover, $s[i]$ is the color of the vertex v_i in Φ_i. Otherwise, $block(i) = 0j$, where $j \in \{j_1, j_2, \ldots, j_r\}$. This means that a monochromatic hyperedge, namely $H_j^{v_i}$, was detected after the vertex v_i was colored in i-th iteration. Since every vertex of $H_j^{v_i}$ other than $v^{\min}(H_j^{v_i})$ gets uncolored during this iteration of the algorithm, one can obtain Φ_{i-1} from Φ_i by coloring every vertex in $V(H_j^{v_i}) \setminus \{v^{\min}(H_j^{v_i})\}$ with the color of the vertex $v^{\min}(H_j^{v_i})$ in Φ_i. Again, $s[i]$ is the color of the vertex $v^{\min}(H_j^{v_i})$ in Φ_i. This completes the proof. □

Corollary 4. *Let s_1 and s_2 be two input strings of length t. If $s_1 \neq s_2$, then $g(s_1) \neq g(s_2)$.*

It is not hard to see that using arguments similar to that in the proof of Lemma 6 in the previous Chapter, we can obtain the same result; so we give the statement without proof.

Lemma 15. $r \leq \dfrac{t}{k-1}$.

Let Z be the total number of possible input strings of length t, so $|Z| = 2^t$. And also suppose that $g(Z)$ be the total number of output strings corresponding to input strings in Z. Recall each output string is of the form $(000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0\Phi)$, where each j_q has D possible values. And Φ can be one among 3^n possiblities. Consider $g^*(s)$ be the string excluding Φ from $g(s)$, and then replacing every non-zero symbol by '1', as follows:

$$g(s) = 000\ldots0j_1000\ldots0j_20\ldots j_r000\ldots0 \underbrace{\Phi}_{\text{is removed}}$$
$$g^*(s) = 000\ldots0\,1\,000\ldots0\,1\,0\ldots1\,000\ldots0$$

Notice that two j_q's cannot be printed consecutively and $block(1)$ is always 0. And recall that algorithm prints exactly t 0's after the termination of the algorithm. So there are $\binom{t}{r}$ many total possible $g^*(s)$'s. Then,

$$|g(Z)| \leq 3^n \binom{t}{r} D^r$$

Therefore, using Lemma 15,

$$|g(Z)| \leq 3^n \binom{t}{\frac{t}{k-1}} D^{\frac{t}{k-1}}$$

Recall that we have assumed that the algorithm does not fully color the graph G for any string in Z. From Corollary 4, we know that $|g(Z)| \geq |Z|$. Thus:

$$\begin{aligned}
3^n \binom{t}{\frac{t}{k-1}} D^{\frac{t}{k-1}} &\geq 2^t \\
3^n \left(\frac{et}{\frac{t}{k-1}}\right)^{\frac{t}{k-1}} D^{\frac{t}{k-1}} &\geq 2^t \text{(using Stirling's approximation)} \\
3^{\frac{n(k-1)}{t}} \cdot e \cdot (k-1) \cdot D \cdot 2^{1-k} &\geq 1
\end{aligned}$$

$$e \cdot (k-1) \cdot D \cdot 2^{1-k} \geq \left(\frac{1}{3}\right)^{\frac{n(k-1)}{t}}$$

Now suppose that $e \cdot (k-1) \cdot D \cdot 2^{1-k} < 1$. Let $\epsilon = 1 - e \cdot (k-1) \cdot D \cdot 2^{1-k}$. Then clearly, $e \cdot (k-1) \cdot D \cdot 2^{1-k} = 1 - \epsilon$ and $\epsilon > 0$. Then from the above inequality, we have

$$\begin{aligned}
\left(\frac{1}{3}\right)^{\frac{n(k-1)}{t}} &\leq 1 - \epsilon \\
\Rightarrow \left(\frac{1}{3}\right)^{n(k-1)} &\leq (1-\epsilon)^t \\
\Rightarrow n\,(k-1) \log \frac{1}{3} &\leq t \log(1-\epsilon) \\
\Rightarrow n\,(k-1) \log(3) &\geq t \log\left(\frac{1}{1-\epsilon}\right) \\
\Rightarrow n\,(k-1)\,\frac{\log(3)}{\log\left(\frac{1}{1-\epsilon}\right)} &\geq t
\end{aligned}$$

This means if $t > n\,(k-1)\,\frac{\log(3)}{\log\left(\frac{1}{1-\epsilon}\right)}$, then it is not possible that the algorithm fails to fully color the graph G for every input string of length t. From this, we can conclude that if $e \cdot (k-1) \cdot D \cdot 2^{1-k} < 1$, then there is some input string for which the algorithm fully colors the graph G; i.e. or in other words, the k-uniform hypergraph G with n vertices and m hyperedges has a 2-edge coloring in which there is no monochromatic hyperedge.

So the inequality becomes

$$\begin{aligned}
e \cdot (k-1) \cdot D \cdot 2^{1-k} &< 1 \\
\Rightarrow D &< \frac{2^{k-1}}{e(k-1)}
\end{aligned}$$

By the LLL 7 we had $D + 1 < \frac{2^{k-1}}{e}$, but our bound $D < \frac{2^{k-1}}{e(k-1)}$ is comparatively weaker. Our future plan is to count $g(Z)$ accurately using *N-Dyck words* (see conclusion) to enhance the bound.

*****

Conclusion and future works

The success of this research opens several avenues for future work. One promising direction is to apply the entropy compression method to other graph-theoretic problems and those related to the Lovász Local Lemma. Potential applications include list coloring, where each vertex in a graph is assigned a list of allowable colors, and edge-disjoint path problems, which have practical implications for routing and network optimization.

A significant challenge identified during this research is the counting of specific combinatorial objects, we defined them as partial N-Dyck words. These are the binary strings in which every prefix contains at least N times as many 0's as 1's. Developing a method to efficiently count these words is crucial for refining our bounds and enhancing our understanding of the underlying combinatorial structures.

Addressing these future research directions will not only refine the current bounds but also broaden the practical and theoretical applications of entropy compression in combinatorial optimization and graph theory. The ongoing development of algorithmic methods for these problems will further contribute to the field, providing deeper insights and more robust solutions to complex combinatorial challenges.

*****

Bibliography

[1] N. Alon, B. Sudakov, and A. Zaks. Acyclic edge colorings of graphs. *Journal of Graph Theory*, 37:157–167, 2001.

[2] R. Bissacot, R. Fernández, A. Procacci, and B. Scoppola. An improvement of the lovász local lemma via cluster expansion. *Combin. Probab. Comput.*, pages 709–719, 2011.

[3] D. Conlon, J. Fox, and B. Sudakov. Recent developments in graph ramsey theory. In Artur Czumaj, Agelos Georgakopoulos, Daniel Král, Vadim Lozin, and Oleg Pikhurko, editors, *Surveys in Combinatorics 2015*, London Mathematical Society Lecture Note Series, pages 49–118. Cambridge University Press, 2015.

[4] R. Diestel. *Graph Theory*. Springer Publishing Company, 5th edition, 2017.

[5] P. Erdős. Some remarks on the theory of graphs. *Bull. Amer. Math.*, pages 292–294, 1947.

[6] L. Esperet and A. Parreau. Acyclic edge-coloring using entropy compression. *European Journal of Combinatorics*, 34:1019–1027, 2013.

[7] J. Fiamčík. The acyclic chromatic class of a graph (in russian). *Mathematica Slovaca*, 28:139–145, 1978.

[8] J. Fiamčík. Acyclic chromatic index of a graph with maximum valency three. *Arch. Math.2, Scripta Fac. Sci, Nat. UJEP Bruensis*, 16:81–87, 1980.

[9] J. Fiamčík. Acyclic chromatic index of a subdivided graph. *Arch. Math.2, Scripta Fac. Sci, Nat. UJEP Bruensis*, 19:69–82, 1984.

[10] F. Guldan. Acyclic chromatic index and linear arboricity of graphs. *Mathematica Slovaca*, 41(1):21–27, 1991.

[11] M. Molloy and B. Reed. Further algorithmic aspects of the local lemma. *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 524–529, 1998.

[12] R. A. Moser and G. Tardos. A constructive proof of the general lovasz local lemma. *ACM journals*, pages 1–15, 2010.

[13] E. Mulligan. Entropy compression. https://fse.studenttheses.ub.rug.nl/id/eprint/27962/, 2022.

[14] C. Mcdiarmid N. Alon and B. Reed. Acyclic coloring of graphs. *Random Structures & Algorithms*, 2:277–288, 1991.

[15] S. Ndreca, A. Procacci, and B. Scoppola. Improved bounds on coloring of graphs. *European J. Combin.*, pages 592–609, 2012.

[16] F. P. Ramsey. On a problem of formal logic. *Proc. London Math. Soc.*, 30:264–286, 1930.

[17] V.G. Vizing. Critical graphs with given chromatic class. *Diskret*, Analiz 5:9–17, 1965.

[18] J. Vondrák. Non-constructive methods in combinatorics. `https://theory.stanford.edu/~jvondrak/MATH233-2016/Math233-lec02.pdf`, 2020.