EPFL

Master semester project

DISOPT

Non-repetitive coloring of graphs

Author:
Gilbert Maystre

Supervisor:
Manuel Aprile

Responsible:
Prof. Friedrich Eisenbrand

June 9, 2017

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Contents

1 Abstract & Notations

1.1 Abstract

In this report, we study an extension of proper graph coloring called non-repetitive graph coloring. After having stated the main definitions, we present a powerful idea introduced by [?] which provides upper-bound to Thue numbers of graphs. We propose a small extension of this result to Thue indices. We then proceed to one of our main research topic which is studying the difference between $\pi(L(T))$ and $\pi'(T)$ for trees. Notably, we show that some classes of tree have their difference bounded by a constant and improve the previous upper-bound for the gap to $2\Delta(T)$ in the general case. To close the theoretical part, we link non-repetitive graph colorings to the concept of treewidth which is currently a hot topic in graph theory. We provide the first upper bound for the Thue index in term of treewidth: $\pi'(G) \in O(\Delta^2 t)$.
Finally, we show how to actually compute non-repetitive graph colorings by going trough other problems (SAT and a LP formulation) and provide a few optimizations idea, such as separate paths. Using our codebase, we verify our conjecture for a large number of trees.

1.2 Notations

We denote by $[k]$ the set $\{1, \ldots, k\}$

In a graph $G = (V, E)$, we use the following neighbourhood notation:

- For $v \in V$, we denote by $\mathcal{N}(v)$ the set of vertices adjacent to v.
- For $e \in E$, we denote by $\mathcal{N}(e)$ the set of edges incident to e.
- For $W \subseteq V$, we set $\mathcal{N}(W) = \bigcup_{v \in W} \mathcal{N}(v)$
- For $M \subseteq E$, we set $\mathcal{N}(M) = \bigcup_{e \in M} \mathcal{N}(e)$
- For $v \in V$, we denote the set of all the vertices at distance k of v by
$$\mathcal{N}^k(v) = \{u \in V : d(u, v) = k\}$$

If S is a sequence, $S(i)$ denotes the element at position i in S.

The line graph $L(G)$ of a graph $G = (V, E)$ is defined as follow: the vertex set of $L(G)$ is E and there is an edge between two vertices in $L(G)$ if their corresponding edges touches in G.

Throughout this paper, the maximum degree of G is denoted $\Delta(G)$ and the threewidth $t(G)$.

Logical false and true are represented by $\mathbf{F}$ and $\mathbf{T}$ respectively.

2 Introduction

2.1 Non-repetitive colorings

A sequence $S = s_1, s_2, s_3, \ldots$ is non-repetitive if it does not contains any square, a finite sequence of the form $x_1, x_2, \ldots, x_{2r}$ where $x_i = x_{i+r}$ for all $i \in [r]$. In 1906, the Norwegian mathematician Axel Thue proved that there exists arbitrarily long non-repetitive sequence using only three symbols [?]. This result is amazing since restricted to two symbols, the longest non-repetitive sequence we can produce is *aba*. The Thue sequence can be constructed starting with a and then applying the following mapping iteratively:

$$a \mapsto abcab$$
$$b \mapsto acabcb$$
$$c \mapsto acbcacb$$

Since then, Thue's result has been generalized in the context of combinatorics on word (see [?], for instance) and a graph theoretic generalization was proposed by Alon & al. in [?]. We define non-repetitive graph coloring as follows:

Definition 1. Let $G = (V, E)$ be a graph and $c : V \to C$ a coloring. If the sequence of colors along any simple path is non-repetitive, then we say that c is **non-repetitive**.

See Figure 1 for an example. We define in the same way non-repetitive coloring of the edges (by taking a coloring $c : E \to C$). At this point, notice that any non-repetitive graph coloring is also a proper coloring because the edges are trivial paths. Following what has been done for proper colorings, we define the Thue number and Thue index as follows:

Definition 2. Let $G = (V, E)$ be a graph. We denote by $\pi(G)$ the minimum number of colors that any non-repetitive coloring must use (the **Thue number**). We denote by $\pi'(G)$ the minimum number of colors that any non-repetitive coloring of the edges must use (the **Thue index**).

Figure 1: On the left, the graph is properly colored but not non-repetitively colored. Using one more color, we can color it non-repetitively (on the right). Furthermore this coloring is optimal and thus $\pi(G) = 4$

We use this notation to match modern papers on the subject but be aware that some early publications used $\pi(G)$ for the Thue index. Thue's result guarantees that any path on n vertices P_n has $\pi(P_n) \leq 3$ (and with equality for $n \geq 4$). Also, the cycle on n vertices C_n has $\pi(C_n) \leq 4$ since we can take a path P_{n+1}, color it non-repetitively with three colors and then glue vertices v_1 and v_{n+1} and give the resulting vertex a new unique color. Because any proper coloring of the complete graph $\mathbb{K}_n$ needs n colors, we have $\pi(\mathbb{K}_n) = n$. This is so, because as stated any non-repetitive coloring is also proper and thus for any graph $\pi(G) \geq \chi(G)$.

2.2 Complexity of non-repetitive graph coloring

Given a graph G, the **Thue number problem** is deciding whether there exists a non-repetitive coloring for G which uses at most k colors. [**?**] notice that this problem belongs to the complexity class $\sum_2^p = \mathsf{NP}^{\mathsf{NP}}$ and conjecture that it is complete for this class. Intuitively, this complexity class in the polynomial hierarchy represents all problems that admit a non-deterministic polynomial time algorithm provided this algorithm has access to an oracle capable of solving problems in NP.

Given a graph $G = (V, E)$ and a coloring $c : V \to C$, checking whether c is repetitive lies in coNP. That is, checking that c is repetitive is in NP. To see that, notice that a proof of repetitiveness of G is simply a path P in G which is repetitively colored under c and this certificate is verifiable in time $O(|V|)$. [**?**] proves that actually, this problem is complete for that class using a reduction from the Hamiltonian path problem.

3 Coloring techniques & levelings

3.1 Coloring locally

The intrinsic difficulty of coloring graphs non-repetitively is that the color of a vertex might be influenced by another one very far in the graph. Indeed, those two vertices might be connected by several paths and any path must be non-repetitive. This kind of difficulty does not arise in proper graph coloring : a vertex is colored legally if all its neighbours have a different color. This "local verifiability" property yields some very elegant proper colorings that are expressed recursively, such as the coloring of trees using two colors but this kind of trick is not directly available in our setting.

In this section, we introduce a very powerful technique introduced by Kündgen & Pelsmajer in [?] that express a way to color a graph non repetitively by only looking at small parts of the graph at a time. This result will be useful in subsequent sections. Finally, we propose an extension to this trick which targets the Thue index.

3.2 Shadow-complete levelling

The trick relies on non-repetitive palindrome-free sequence. We say that a sequence $X = x_1, x_2, \ldots, x_{2r+1}$ is a palindrome if $X = x_{2r+1}, \ldots, x_2, x_1$ and a sequence S is palindrome-free if it doesn't contain any palindrome. As mentioned in [?] there exists arbitrarily long non-repetitive palindrome-free sequences. It suffices to take a non-repetitive sequence on a, b and c and insert the new symbol d every two symbols. We now move on defining level partitions.

Definition 3. Let $G = (V, E)$ be a graph and let $\mathcal{V} = \{V_i\}_{i=1}^m$ be a partition of the vertices. We say that $\mathcal{V}$ forms a **level partition** of G if $\mathcal{N}(V_i) \subseteq V_{i-1} \cup V_i \cup V_{i+1}$ for all $i \in [m]$. We denote by respectively G_k and $G_{>k}$ the subgraphs of G induced by V_k and $\bigcup_{j>k} V_j$ respectively.

The idea of their paper is to color a graph in two passes. Given a graph $G = (V, E)$, first color each level separately. This yields a coloring $c_l : V \to C$. Then, given a non-repetitive and palindrome-free sequence S, we build the final coloring c such that if v is in level i, $c(v) = (S(i), c_l(i))$. To prove that c is non-repetitive, we need the levelling to have an additional property.

Definition 4. Let $\mathcal{V} = \{V_i\}_{i=1}^k$ be a level partition of $G = (V, E)$. The k**-shadow** of a subgraph $H \subseteq G$ is the set of vertices in V_k which have a neighbour in $V(H)$. We say that G is shadow complete (with respect to $\mathcal{V}$) if the k-shadow of every component of $G_{>k}$ induces a complete graph.

It turns out that having this shadow-completeness property is enough to prove that the coloring described above is non-repetitive as the following states:

Theorem 5. [?] If G has a shadow-complete level partition $\mathcal{V}$ then

$$\pi(G) \leq 4 \max_{i \in [k]} \{\pi(G_K)\}$$

Actually, this upper bound can be improved by a tighter analysis. If the non-repetitive palindrome-free sequence S used to color the levels is on the alphabet Γ, then we propose the following:

$$\pi(G) \leq \sum_{\gamma \in \Gamma} \max_{k \colon S(k)=\gamma} \{\pi(G_k)\} \tag{1}$$

We will use this finer bound in section 4.2. The next problem is how to come up with a shadow-complete level partition. Kündgen & Pelsmajer give an example for chordal graphs. Recall that a graph is chordal if it contains no induced cycle on 4 or more vertices.

Lemma 6. [?] Let G be a connected chordal graph with clique number $\omega > 1$ and let x be any vertex. G is shadow complete with respect to the partition given by $V_k = \mathcal{N}^k(x)$ and every G_k is a chordal graph with clique number $< \omega$

3.2.1 Example of application: coloring $L(T_k)$

We will describe a very nice application of the previous section that we have used to give the first improvement to the small gap conjecture. As will be motivated in section 4, upper bounding the Thue number of the line graph of the complete k-ary tree is crucial to lower bound the gap.

Lemma 7. Let T_k be a complete k-ary tree of any height. Then, $\pi(L(T_k)) \leq 4k$

Proof. Notice that $L(T_k)$ is a block tree where each block is a clique of size $k+1$ or k (for the central block). Therefore, $L(T_k)$ has no induced cycle on 4 vertices or more, in other words, $L(T_k)$ is a chordal graph. So, from Lemma 6, the level partition given by $V_k = \mathcal{N}^k(v)$ where v is any vertex is shadow-complete. Notice that each V_k is a collection of disconnected cliques of size at most k therefore each V_k is k-colorable. From theorem 5, we conclude that $\pi(L(T_k)) \leq 4k$. □

Reader familiar with the concepts of section 4 might recognize that this shows that for any tree T, $\pi(L(T)) - \pi'(T) \leq 3\Delta - 4$.

3.3 Star levelling

A natural follow-up question to be asked is how to extend this result to the Thue index. To do this, we are going to need a kind of levelling for the edges which allow us to use a trick similar the one of [?]. We first propose a straightforward edge levelling definition.

Definition 8. Let $G = (V, E)$ be a graph and $\mathcal{E} = \{E_i\}_{i=1}^m$ a partition of its edges. We say that $\mathcal{E}$ is an **edge-levelling** if $\mathcal{N}(E_i) \subseteq E_{i-1} \cup E_i \cup E_{i+1}$ for all $i \in [m]$. Furthermore, define by G_k and $G_{>i}$ the graphs induced by the edges in E_k and $\bigcup_{i>k} E_i$ respectively.

Definition 9. Let $\mathcal{E}$ be an edge-levelling of $G = (V, E)$. We say that $\mathcal{E}$ is a **star levelling** if the k-shadow of every component of $G_{>k}$ induces a star graph.

At this point, we can state the following result:

Theorem 10. Let $\mathcal{E} = \{E_i\}_{i=1}^m$ be a star levelling of a graph G. Then (without tighter analysis)

$$\pi'(G) \leq 4 \max_{i \in [m]} \{\pi(L(G_i))\}$$

Proof. The line graph of star is a complete graph thus in the line graph domain, $\mathcal{E}$ is shadow-complete. Using the fact that $\pi'(G) \leq \pi(L(G))$ (section 4.1) we use Theorem 5 to conclude. $\square$

Notice that this bound might not be optimal. Indeed, the difference between $\pi(L(G))$ and $\pi'(G)$ can be made arbitrarily large (see section 4.1).

4 Small gap conjecture

4.1 Coloring the line graph

Given a graph $G = (V, E)$, its line graph $L(G)$ is defined as follows: The vertex set of $L(G)$ is E and there is an edge between two vertices if their corresponding edges touches in G. In term of proper colorings, coloring the edges of a graph is exactly the same problem as coloring the vertices of the line graph and we have $\chi'(G) = \chi(L(G))$. In the setting of non-repetitive graph coloring, the only thing we have is $\pi'(G) \leq \pi(L(G))$ [**?**]. It is due to the fact that any path in a graph is also a path in the line graph, but the converse is not necessarily true. Aprile proved that the difference between $\pi(L(G))$ and $\pi'(G)$ can be made arbitrarily large using complete graphs [**?**]. However, it is still an open question to know whether the ratio

$$\frac{\pi(L(G))}{\pi'(G)}$$

is unbounded. After several attempts, we were not able to provide an answer to this question. If the general case is unclear, there exists some classes of graph for which the two quantities are equal e.g. paths, cycle and star graphs. It is conjectured that for trees, the difference between $\pi(L(T))$ and $\pi'(T)$ is bounded by a constant d. Throughout this report, we will call this conjecture the *small gap conjecture*. For the moment, all the trees we have tested satisfy this conjecture with d at most 1 (see section 6.4 for details).

4.2 Classes of tree for which the conjecture holds

Apart from paths and star graphs, Amendt proved that the conjecture also holds for caterpillars [**?**]. Caterpillars are trees in which all the vertices are within distance 1 of a central path (see Figure 2 for an example of caterpillar). This result will be exploited in section 6.3.3 to speed up our conjecture checker. We reformulate the proof using the framework of section 3.2, illustrating its power.

Theorem 11. [**?**] For any caterpillar T, we have that $\pi(L(T)) - \pi'(T) \leq 1$

Proof. Let Δ be the degree of T. The line graph of T consist of the main path and cliques attached to this path. Consider the levelling of the vertices defined by putting all the vertices of the main path in V_1 and all the others in V_2. Notice that the 1-shadow of every component of V_2 is simply an edge which is a clique. Therefore, this levelling is shadow-complete. Since V_1 induces a path, three colors are needed for the first level. V_2 consists of a cloud of cliques of size at most $\Delta - 2$ therefore, $\Delta - 2$ colors are needed for the second level. Notice that we can make a non-repetitive palindrome-free sequence of length 2 using two colors: *ab*. Thus, using the finer bound (Equation 1), we can conclude that $\pi(L(T)) \leq \Delta + 1$. We finish the proof by noticing that $\pi'(T) \geq \Delta$ □

We have found two other classes for which this conjecture holds: subdivisions of star and subdivisions of caterpillar.

Definition 12. [?] In a graph $G = (V, E)$ a subdivision of an edge $uv \in E$ is the operation of replacing uv with a path u, w, v trough a new vertex w. A **subdivision of** G is a graph obtained from G by successive edge subdivision.

Theorem 13. If T is a subdivision of a star, then we have that $\pi(L(T)) - \pi'(T) \leq 1$

Proof. Let Δ be the degree of the central vertex. T is a star with edge extended in paths therefore its line graph is a clique of degree Δ where each vertex of the clique is extended in a path. Take two vertices in the Δ-clique and color both their extended path and the connecting edge non-repetitively with three colors a, b and c. Color all other $\Delta - 2$ vertices in the clique with $\Delta - 2$ unique colors. Remaining paths are colored using the colors of the first path (a, b and c). This coloring uses $\Delta + 1$ colors and we conclude the claim by noticing that $\pi'(T) \geq \Delta$ □

Theorem 14. If T is a subdivision of a caterpillar, then we have that $\pi(L(T)) - \pi'(T) \leq 2$

Proof. Let Δ be the max degree. Notice that $L(T)$ is a main path plus a cloud of subdivision of stars attached to it. Color the main path with three colors and all the subdivisions of star with $\Delta - 1$ color as described in previous theorem. Notice that this separation is shadow complete with respect to the main path. Thus, no repetitive path shall exist. □

For general trees, no such bound is known yet. The previous best bound stated that $\pi(L(T)) - \pi'(T) \leq 3\Delta(T) - 3$ [?]. This bound is the result of a greedy algorithm. For a straightforward (yet very small) improvement of this bound, see section 3.2.1. The goal of this section is to prove a 2Δ upper-bound. To do so, we extend a paper from Kündgen & Talbot [?] which studies the Thue index of trees.

Figure 2: On the left, a caterpillar with edges colored non-repetitively. On the right the line graph of the same caterpillar colored non-repetitively. Both colorings are optimal and $\pi(L(T)) = \pi'(T) + 1$ so the bound of Theorem 11 is tight. Example found by Aprile [?].

4.3 Coloring the edges of a tree

It is known that any tree has Thue number equal to 4 (this can be proved using the result of section 3.2). However, the question is still not settled for the Thue index of trees. Another simple application of section 3.2 yields the upper-bound $\pi'(T) \leq 4(\Delta(T) - 1)$, but it is very far from the lower bound $\Delta(T)$. In [?] Kündgen & Talbot show that $\pi'(T) \leq 3(\Delta(T) - 2)$. We will recall almost all of their definitions and technical lemmas since we are going to need them for our own result.

4.3.1 Complete k-ary trees

To ease the notation and the proof, we work with complete k-ary trees. A k-ary tree is a tree with designated root and the property that every vertex that is not a leaf has exactly k children. The k-ary tree in which the distance from the root to every leaf is h is denoted by $T_{k,h}$. Most of the time, the height will not be relevant and we will simply write T_k. We give each edge of T_k an index $i \in \mathbb{N}$ in the following manner (see Figure 3 for an example):

1. The edges adjacent to the root get indices 1, 2, ..., k
2. All the edges adjacent (in the lower level) to an edge indexed i get indices $i+1$, $i+2$, ..., $i+k$.

Notice that the same index might appear several times. This indexing will help us convert a flat color sequence into a coloring of the edges of T_k.

Definition 15. Let $S = s_1, s_2, \ldots$ be a sequence of colors. The edge-coloring of T_k **derived** from S is the coloring where any edge with index i gets color s_i (see Figure 4 for an example).

Figure 3: On the left, lies a summary of how to indexes the edges in T_k. On the right, an example of such indexing for $T_{2,2}$

4.3.2 k-special sequences

At this point, we need to characterize which sequence of colors S yields a non-repetitive coloring. To do so, the following definition was proposed:

Definition 16. [?] Let $S = s_1, s_2, \ldots$ be a (finite or infinite) sequence. A sequence of indices $i_1, i_2, \ldots, i_{2r}$ is called k**-bad** for S if there is an m with $1 < m \leq 2r$ such that the following four conditions hold:

1. $s_{i_1}, s_{i_2}, \ldots, s_{i_{2r}}$ is a repetition
2. $i_1 > i_2 > \cdots > i_m < i_{m+1} < i_{m+2} < \cdots < i_{2r}$
3. $|i_j - i_{j+1}| \leq k$ for all j with $1 \leq j < 2r$
4. $i_{m+1} < i_m + k$ if $m < 2r$

Furthermore, S is called **k-special** if it has no k-bad sequence of indices.

Note that item 2, 3, and 4 force the sequence of indices to correspond to a path in T_k (recall how we defined the indexing of T_k). The following theorem justifies this definition:

Theorem 17. [**?**] An infinite sequence S is k-special if and only if the edge-coloring of T_k derived from S is non-repetitive

In the proof, only the backward implication needs the fact that S is infinite (the interested reader can refer to the original paper [**?**]). Remark the beauty of their work: at this point, the initial problem (coloring the edges of T_k non-repetitively) has been completely abstracted to the other problem of finding k-special sequences using few colors. Kündgen & Talbot actually show how to build a k-special sequence using $3k+1$ colors, which implies that $\pi'(T) \leq 3\Delta(T) - 2$. However, we will describe another sequence S_k which uses $3k+3$ colors. Although not optimal, S_k has some additional properties that will be needed for our result.

Definition 18. Let S be an infinite non-repetitive sequence on a, b and c. Then, for $k \geq 2$, we call S_k the sequence obtained from S by replacing each a, b and c by a block of $k+1$ colors:

$$\begin{aligned} a &\mapsto a^{(1)}\, a^{(2)}\, \ldots\, a^{(k+1)} \\ b &\mapsto b^{(1)}\, b^{(2)}\, \ldots\, b^{(k+1)} \\ c &\mapsto c^{(1)}\, c^{(2)}\, \ldots\, c^{(k+1)} \end{aligned}$$

Lemma 19. [**?**] For $k \geq 2$, S_k as defined above is k-special and uses $3k+3$ colors.

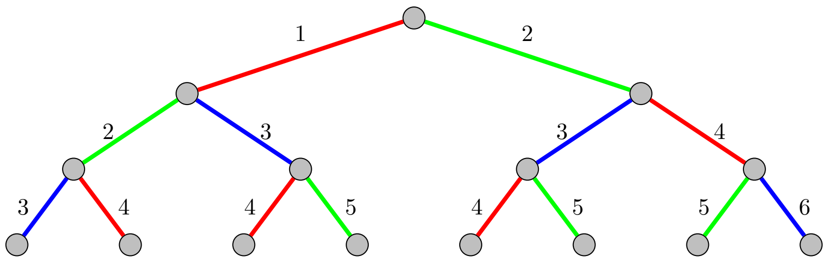

Figure 4: the coloring of $T_{2,3}$ derived from the sequence $S = r, g, b, r, \ldots$. Notice that this edge coloring is repetitive, thus S cannot be k-special.

4.4 Improvement for the small gap conjecture

4.4.1 Coloring the vertices of $L(T_k)$

In this section, we show how to use the edge coloring of T_k derived from S_k to color the vertices of $L(T_k)$ non-repetitively. More precisely, we define the derived coloring of $L(T_k)$ as follows:

Definition 20. Each vertex of $L(T_k)$ get the index of its corresponding edge in T_k. We then define the vertex-coloring c_{S_k} of $L(T_k)$ **derived from** S_k as follows: every vertex of $L(T_k)$ get the color of its corresponding edge in T_k with coloring derived from S_k. In other words, if v has index i, then $c_{S_k}(v) = S_k(i)$.

Notice that the fact that this coloring is non-repetitive is quite lucky: In general, constructing a non-repetitive coloring of $L(G)$ is harder than constructing an edge non-repetitive coloring of G, as mentionned in section 4.1. We decided to try only after a large amount of computer simulations. Finally, we managed to come up with a proof of the following claim:

Theorem 21. The coloring c_{S_k} is non-repetitive and thus $\pi(L(T_k)) \leq 3k + 3$

To prove this, we are not going to work with $L(T_k)$ directly but rather with sequences of indices. Thus, we need to characterize what sequence of indices can represent legal paths in $L(T_k)$.

Definition 22. If a sequence of indices $I = i_1, \ldots, i_{2r}$ is the result of a valid path in $L(T_k)$, we say that I is an **honest sequence**.

We now show that honest sequences have a lot in common with sequences of indices that can appear as the result of a path in T_k (axioms 2-3-4 of Definition 16)

Lemma 23. For any honest sequence $I = i_1, \ldots, i_{2r}$, the following hold:

1. $|i_j - i_{j+1}| \leq k$ for all $1 \leq j < 2r$
2. There exists an unique global minimum index m (i.e. an unique $m \in [2r]$ such that $i_m = \min_{j \in [2r]}\{i_j\}$)

Proof. The first item is clear from the way we index our vertices. To prove the second let's assume for contradiction's sake that there exists two global minimum indices i_m and $i_{m'}$ with $m \neq m'$. Notice that any children of i_m is bigger than i_m and the same reasoning applies to $i_{m'}$, so since they are both minima, i_m and i'_m share their parent but then this situation is impossible since each child has a different index. □

Lemma 24. Let $I = i_1, \ldots, i_{2r}$ be an honest sequence with unique minima m. Then if $m < 2r$, we have that $i_{m+1} < i_m + k$ or $i_{m-1} < i_m + k$. In other words, if I doesn't satisfy axiom 3, then the reverse of I must satisfy it.

Proof. Let $P = v_1, v_2, \ldots, v_{2r}$ be the path in $L(T_k)$ that generated I. For sake of contradiction, let us assume that $i_{m+1} = i_m + k$ and $i_{m-1} = i_m + k$ (it cannot be greater than k because of Lemma 23). The last two equalities implies that v_{m-1} is the farthest left child of v_m and also that v_{m+1} is the farthest left child of v_m too, therefore P is not a valid path, a contradiction with the fact that I is honest. □

At this point, it should be noted that in general, axiom 2 of Definition 16 is not satisfied by honest sequences. For instance, the sequence of indices $3, 2, 3, 1$ is not of the described form. We formalize this with the following definition.

Definition 25. Let $I = i_1, \ldots, i_{2r}$ be an honest sequence. We say that $w \in \{2, \ldots, 2r-1\}$ is a **wiggle point** if we have $i_{w-1} < i_w > i_{w+1}$. The **dual** of a wiggle point w is the position $w + r$ if $w \leq r$ and $w - r$ else. We say that a wiggle w point is **removable** if its dual is also a wiggle point or an extremity (that is, $w = r$ or $w = r + 1$).

Lemma 26. If w is a wiggle point in an honest sequence $I = i_1, \ldots, i_{2r}$, then

$$|i_{w-1} - i_{w+1}| \leq k$$

Proof. Since w is a wiggle point, we have $i_w > i_{w-1}$ and $i_w > i_{w+1}$. It follows that i_{w-1} and i_{w+1} cannot be children of i_w so they must lie at the same level as i_w or at most one can be the parent of i_w. Therefore i_{w-1} and i_{w+1} cannot be more than k apart. □

To deal with wiggle points, we will need an additional properties of S_k. We state and prove it now.

Lemma 27. [order preservation] Let S_k be defined as before. Then for any choice of four indices $i_1 \neq i_2$ and $i_3 \neq i_4$ such that:

1. $|i_1 - i_2| \leq k$ and $|i_3 - i_4| \leq k$
2. $S(i_1) = S(i_3)$ and $S(i_2) = S(i_4)$

the order is preserved: $i_1 < i_2 \iff i_3 < i_4$

Proof. Proving only one direction will be enough since the other is symmetric. Following the notation used when building S_k, let $S(i_1) = s^{(x)}$ and $S(i_2) = t^{(y)}$. If $s = t$, the claim is straightforward since $x \leq y$ and we cannot have two times the same symbol contiguously in the initial sequence. If $s \neq t$, we have to check that if at some point in the initial sequence t happens just before s, then the two would be too far apart to be in the same window. First, notice that $x > y$, else we cannot have $|i_1 - i_2| \leq k$. Now, if t happens before s, then the distance between $t^{(y)}$ and $s^{(x)}$ would be $k + 1 - y + x$ but with our previous remark, this is at least $k + 2$. Hence in a window of size k, we cannot have $t^{(y)}$ before $s^{(x)}$. □

We are now ready and set up to prove the main claim.

Lemma 28. Let $I = i_1, \ldots, i_{2r}$ be an honest sequence, then I cannot be repetitively colored under c_{S_k}.

Proof. By induction on $r \geq 1$. For $r = 1$, we have $I = \alpha, \beta$ and by Lemma 23 $|\alpha - \beta| \leq k$. Furthermore, I doesn't have enough indices to contain a wiggle point and I is either of the form $\alpha < \beta$ or $\alpha > \beta$. Thus, if I is repetitively colored, then I is a k-bad sequence, a condtraidiction with the fact that S_k is k-special.
For the induction step, let's suppose for the sake of contradiction that I is repetitively colored under c_{S_k}. Since I is an honest sequence, I has an unique minima m and $|i_j - i_{j+1}| \leq k$ for all $j \in [2r - 1]$ (Lemma 23). We derive a contradiction for each of the

following cases, which cover all possibilities.

<u>Case 1:</u> I contains an non-removable wiggle point w. Assume w.l.o.g that w appears in the first half ($w < r$). Then, from the order property of S_k (Lemma 27), we must have the following situation:

$$i_1 \dots i_{w-1} < i_w > i_{w+1} \dots i_r \,|\, i_{r+1} \dots i_{w-1+r} < i_{w+r} > i_{w+1+r} \dots i_{2r}$$

So $w + r$ is also a wiggle point, a contradiction with the fact that w is non-removable.

<u>Case 2:</u> I contains a removable wiggle point w. Notice that because of Lemma 26 removing w and its dual yields another honest sequence of smaller size which is still repetitive, a contradiction with the induction hypothesis.

<u>Case 3:</u> I contains no wiggle points. If I is not such that $m < 2r \implies i_{m+1} < i_m + k$, consider the sequence I' obtained by reversing I. I' must also be repetitively colored, is also honest (we are just following the path defined by I in the other direction) and Lemma 24 guarantee that it's turning point m is such that $m < 2r \implies i_{m+1} < i_m + k$. At this point, I (or I') satisfies all the four axioms of k-bad sequences a contradiction with the fact that S_k is k-special. □

Corollary 29. $\pi(L(T_k)) \leq 3k + 3$

Proof. We've just proved that any honest sequence is non-repetitively colored. It follows that any path in $L(T_k)$ is non-repetitively colored under c_{S_k} too. Since S_k uses $3k + 3$ colors, we can conclude that $\pi(L(T_k)) \leq 3k + 3$. □

4.4.2 Wrapping everything together

It is now easy to conclude the following from Corollary 29.

Theorem 30. Let T be a tree of degree Δ. Then $\pi(L(T)) - \pi'(T) \leq 2\Delta$.

Proof. Notice that T is a subtree of a $(\Delta - 1)$-ary tree of sufficient height. To see that, hang T from any vertex (it will be the root) and complete it. Since any non-repetitive coloring of a graph is also non-repetitive for all of its subgraphs, we have that $\pi(L(T)) \leq \pi(L(T_{\Delta-1}))$. Using Corollary 29, we infer that $\pi(L(T)) \leq 3\Delta$. Now, using the fact that any non-repetitive edge coloring of T must use at the very least use Δ colors (think about the star graph), we have that $\pi'(T) \geq \Delta$. Thus we shall conclude that

$$\pi(L(T)) - \pi'(T) \leq 2\Delta$$

□

5 Coloring and treewidth

5.1 Basic notions of treewidth

The threewidth of a graph is a measure of how far is the graph from being a tree. The rationale behind it is that many combinatorial optimization problems are hard on general graphs but easy on tree e.g. non-repetitive graph coloring for the vertices. Therefore, a graph looking very much like a tree might provide better running time guarantees. For a survey of those kind of application see Bodlaender & Koster [**?**].

Definition 31. [Tree decomposition] A **tree decomposition** of a graph $G = (V, E)$ is a pair $(\mathcal{X} = \{X_i : i \in I\}, \mathcal{T} = (I, F))$ where each node $i \in I$ has associated to it a subset of vertices $X_i \subseteq V$ called the **bag** of i such that:

1. Each vertex belongs to at least one bag: $\bigcup_{i \in I} X_i = V$
2. For all edges, there is a bag containing both its end-points, i.e. for all $uv \in E$, there exists an $i \in I$ with $u, v \in X_i$
3. For all vertices $v \in V$, the set of nodes $\{i \in I : v \in X_i\}$ induces a subtree of $\mathcal{T}$

The width of a tree decomposition is the size of its biggest bag. The **treewidth** of G is the smallest width over all valid tree decomposition minus 1.

Notice that axiom 3 guarantees that $\mathcal{T}$ is a tree. The minus 1 in the definition of treewidth is for convenience only: in this manner, trees have treewidth 1. For an example of tree decomposition, see Figure 5.

Figure 5: A graph G and one of its tree-decomposition of size 3.

5.2 Upper bounds in term of the treewidth

Kündgen & Pelsmajer [**?**] were the first to come up with an upper bound of the Thue number of a graph using its treewidth. As the proof lacks some step in the original paper, we complete it here.

Theorem 32. [**?**] Every graph $G = (V, E)$ of treewidth t has a non-repetitive 4^t-coloring.

Proof. By proposition 12.3.12 of [?], every graph of treewidth t is a subgraph of a chordal graph with clique number $t+1$, so it suffices to prove by induction that chordal graphs with clique number $t+1$ have a non-repetitive 4^t-colorings. For the base step, $t = 0$, observe that chordal graphs with clique number 1 are edgeless and thus have non-repetitive 1-coloring.
For the inductive step, notice that a chordal graph with clique number $(j+1)+1$, has a shadow complete levelling where each level is again a chordal graph and has clique number at most $j+1$ (Lemma 6 of section 3.2). By the induction hypothesis, the levels can thus be colored with 4^j colors. Now, Theorem 5 of section 3.2 guarantees that our chordal graph with clique number $(j+1)+1$ has Thue number 4×4^j therefore it is 4^{j+1}-colorable. □

Note that this bound is tight for trees (recall that trees have treewidth 1) but the question of the existence of a poly(t) bound is still open. Following this paper, Barát & Wood [?] have proposed to include the maximum degree of G to derive another upper-bound:

Theorem 33. [?] For any graph G of treewidth t and maximum degree Δ, we have $\pi(G) \in O(t\Delta)$

At this point, we may wonder whether such a bound can be made for the Thue index. Notice first that any bound must include the maximum degree Δ since the star graph has treewidth 1 (it is a tree) but needs Δ colors (one for each edge) - Thus, there is no hope for a bound depending only on t. We propose here a first simple upper bound of this kind.

Theorem 34. For any graph G of treewidth t and maximum degree Δ, we have $\pi'(G) \in O(t\Delta^2)$.

The intuition is to go trough the line graph and use Theorem 33 to conclude. The central lemma is the following due to Harvey & Wood, which relates the treewidth of a graph and the treewidth of its line graph.

Theorem 35. [?] For any graph G of treewidth t and maximum degree Δ, we have

$$t(L(G)) \leq (t+1)\Delta(G) - 1$$

Lemma 36. Let G be a graph of max-degree Δ. Then, $\Delta(L(G)) \leq 2\Delta - 2$

Proof. Let $e \in E(G)$ be the edge which is connected to the most other edge. It can touch at most $2\Delta - 2$ other edges and it's degree in the line graph is thus at most $2\Delta - 2$. □

We are now able to give a proof of Theorem 34. Indeed, from section 4.1 we know that $\pi'(G) \leq \pi(L(G))$ but from Theorem 33, $\pi(L(G)) \in O(t(L(G))\Delta(L(G)))$. Using the last two lemmas, we conclude that $\pi(L(G)) \in O(t\Delta^2)$ which finishes the claim. □

6 Computing non-repetitive graph colorings

One of the goal of the project was to create a codebase that would compute non-repetitive coloring. This is a great tool to finds quick counter-examples and we have used it to guide our theoretical research during the whole project. For instance, it was very useful to convince ourselves that S_k had good chances of coloring $L(T_k)$ non-repetitively (see section 4.4.1 for details).
As stated in section 2.2, finding a non-repetitive coloring is a hard task and instead of trying a brute-force algorithm or a randomized one (see [**?**] for an implementation of the Moser-Tardos algorithm), we decided to use reductions to other problems with black box solvers available on the market. The next two sections describe our linear programming formulation and a reduction to SAT.

6.1 Linear programming formulation

In the linear programming setting, we try to minimize or maximize a linear function subject to a number of linear constraints. If in addition we want some variable to be integral, the resulting linear program is called a mixed integer program. Those are the hardest to solve but have the advantage of having a richer semantic.

Let $G = (V, E)$ be the graph we want to color in a non-repetitive way and let C be the set of available colors. The size of C should be sufficient e.g. $|C| = |V|$ but having a better upper-bound can greatly improve the running time. For all $v \in V$ and $c \in C$, we introduce the binary variable X_{vc}. It indicates whether color c is used at vertex v or not. Also, for all $c \in C$, we introduce the variable Y_c which indicates if color c is used at all. The goal is indeed to have a minimum number of those Y_c non-zero. Lastly, for any unordered pair of vertices $v_1, v_2 \in V$, we introduce the variable $Z_{v_1v_2}$ which will be set to 1 if v_1 and v_2 have different colors and 2 else. With all those variables, we are now ready to formulate our linear program:

$$\begin{aligned}
\min \quad & \sum_{c\in C} Y_c \\
\text{st.} \quad & Y_c \geq X_{cv} \quad \forall v \in V, \forall c \in C \\
& Z_{v_1v_2} \geq X_{v_1c} + X_{v_2c} \quad \forall c \in C, \forall \{v_1, v_2\} \subseteq V && \text{(2a)} \\
& \sum_{c\in C} X_{vc} = 1 \quad \forall v \in V && \text{(2b)} \\
& \sum_{i=0}^{l-1} Z_{v_iv_{i+l}} \leq 2l - 1 \quad \text{For all even-length paths in } G && \text{(2c)}
\end{aligned}$$

Note that constraint 2a is equivalent to $Z_{v_1,v_2} \geq \max_{c\in C}\{X_{v_1c}+X_{v_2c}\}$ and thus, $Z_{v_1v_2} =$

2 if and only if v_1 and v_2 have different colors. Constraint 2b ensure that exactly one color is given to each vertex and finally constraint 2c forces the coloring to be non-repetitive. Indeed, let $v_1, \dots v_{2l}$ be a simple path in G, then if

$$\sum_{i=0}^{l-1} Z_{v_i v_{i+l}} \leq 2l - 1$$

it implies that at least one $Z_{v_i v_{i+l}}$ is not two and so this particular pair of vertices must have a different color implying that the path is non-repetitive.

6.2 Reduction to SAT

The following reduction to SAT has the advantage of being more natural, but it suffers from the lack of straightforward optimization mechanism. We have to "guess" the minimum number of colors needed and then employ the SAT formulation to check if there exists a non-repetitive coloring with this number of colors. In practice the minimum can be found using a binary search with the cost of adding an $O(\log |V|)$ factor to the overall complexity.

Let $G = (V, E)$ be the graph we want to color in a non-repetitive way and C be the set of available colors. Then, for all $v \in V$, and $c \in C$, we introduce the (boolean) variable X_{vc} which indicates whether color c is used at vertex v. For any unordered pair of vertices $v_1, v_2 \in V$, we introduce the variable $Z_{v_1 v_2}$ which is set to 1 if and only if v_1 and v_2 have the same color. We are now ready to state the constraints to be satisfied:

$$Z_{v_1 v_2} = \bigvee_{c \in C} X_{v_1 c} \wedge X_{v_2 c}$$

$$\bigvee_{c \in C} X_{vc} \quad \forall v \in V \tag{3a}$$

$$\neg(X_{vc_1} \wedge X_{vc_2}) \quad \forall \{c_1, c_2\} \subseteq C, \forall v \in V \tag{3b}$$

$$\neg \bigwedge_{i=0}^{l-1} Z_{v_i v_{i+l}} \quad \text{For all odd-length paths in } G \tag{3c}$$

Constraints 3a and 3b ensures that every vertex is gets exactly one color. Constraint 3c check that all path are non-repetitively colored. Indeed, if a path $v_1, \dots, v_{2l}$ in G is such that

$$\neg \bigwedge_{i=0}^{l-1} Z_{v_i v_{i+l}} = \mathbf{T}$$

then at least one of the $Z_{v_i v_{i+l}}$ is $\mathbf{F}$ and it means in turn that the corresponding pair of vertices have different colors, hence the path is non-repetitively colored.

6.3 Practical implementation

Our codebase works on graphs provided by `EasyGraph`[1]. The transformation from a non-repetitive problem instance to an LP or a SAT instance is performed in `Python`. To solve LPs, we have used `Gurobi`[2], a good LP solver which has the advantage of offering an easy `Python` interface. For SAT instances, we have used Microsoft's `Z3` library[3].
In practice, the LP method was faster than the SAT one, so we preferred the first one and all our results are based on this. While solving the model is indeed the bulk of the work, crafting it is also quite costly as it implies to discover every simple path in G. In the next sections, we describe several way to improve the running time.

6.3.1 Symmetry breaking

An optimization that can have a dramatic impact on performance is pre-coloring large cliques. Indeed, all vertices in a clique must have different colors, so we can directly assign arbitrary (different) colors to those particular vertices. For a graph $G = (V, E)$ and a clique $\{v_1, v_2, \dots, v_k\} \subseteq V$, we add the following constraints to our LP model (and remove the superfluous ones):

$$X_{v_i c} = \begin{cases} 1 & \text{if } c = i \\ 0 & \text{else} \end{cases} \qquad \text{for } i \in [k]$$

To have the most impact, we want to find large cliques. Finding the largest clique in a graph is NP-hard, but we can still hope that a maximal clique will be close to the maximum one. To do so, we select an edge at random, grow it so that it forms a maximal clique and restart this process a few times. A possible heuristic would be to choose the starting edge with probability proportional to its degree.
This optimization is most powerful in dense graphs, with the extreme being the complete graph $\mathbb{K}_n$ where the trick basically solves the model without submitting it to a solver.

6.3.2 Finding all paths in G

To solve this problem, we have experimented several different methods. The final and fastest version uses a nice property noticed by Aprile:

Definition 37. A path $P = v_1, \dots, v_{2l}$ with $l \geq 1$ in a graph is said to be **separate** if no vertex in the first half of the path is adjacent to a vertex in the second half except possibly v_1, v_{2l} and v_l, v_{l+1}.

[1] This application is a simple graphical interface to manage graphs and is the result of some personal project of the author. The code and the application is available at `github.com/DaiSijie/EasyGraph`
[2] `gurobi.com`
[3] `github.com/Z3Prover/z3`

Theorem 38. [?] A coloring of a graph G is non-repetitive if and only if any separate path of G is non-repetitive.

The last theorem allow us to reduce greatly the number of constraints of the LP in some cases. Indeed, we just need to check non-repetitiveness for all separate paths. This leaves us to the problem of finding all separate paths in a graph and we propose an iterative procedure described in detail in Algorithm 1.

Algorithm 1: AllSeparatePaths(G)

Input : A graph $G = (V, E)$
Output: The set $\mathcal{P}$ of all separate paths of G
1 $\mathcal{P} \leftarrow E$
2 $\mathcal{Q} \leftarrow$ a queue initialized with all elements of E
3 **while** $\mathcal{Q} \neq \emptyset$ **do**
4 $\quad \langle v_1\, v_2\, \ldots\, v_{2l} \rangle \leftarrow \mathcal{Q}$.pop()
5 $\quad$ **foreach** $s, t \in V$ such that $\langle s\, v_1\, v_2\, \ldots\, v_{2l}\, t \rangle$ is separate in G **do**
6 $\quad\quad \mathcal{P}$.add($\langle s\, v_1\, v_2\, \ldots\, v_{2l}\, t \rangle$)
7 $\quad\quad$ **if** $s \notin \mathcal{N}(t)$ **then**
8 $\quad\quad\quad \mathcal{Q}$.add($\langle s\, v_1\, v_2\, \ldots\, v_{2l}\, t \rangle$)
9 $\quad\quad$ **end**
10 $\quad$ **end**
11 **end**
12 **return** $\mathcal{P}$

Notice that any path in $\mathcal{Q}$ is separate with the additional property of having v_1 and v_{2l} not neighbours. Thus the check of being a separate path (Line 5) can be done in linear time. Another nice feature of this algorithm is that we can generate all separate paths in ascending order in a lazy fashion. For instance, it is useful when we want to have a rough (but fast) lower-bound on the Thue number of some graph: just generate small separate paths of G and solve the LP for them. We have been using this to compute lower bounds on the Thue number of the rook graph $R_{n \times n}$, a crucial graph in the bounded treewidth conjecture.

6.3.3 Optimizations & heuristics for the small gap conjecture

For details and definition about the small gap conjecture, please refer to section 2. To test the small gap conjecture (and potentially find a counter-example), we have methodically tested a large number of tree. Finding the Thue index of a tree with small number of vertices n is not too costly because the number of separate paths is bounded by n^2, however the number of paths might explode for its line graph. Therefore, the goal was to avoid solving this LP. Here are the three scheme we implemented specially for this conjecture:

- Detection of caterpillar: A tree can be recognized as a caterpillar in linear time and since the conjecture is proved for this class (see section 4.2), there is no need to check it.
- Maximum cliques: On the contrary of general graphs, it is possible to find the largest clique of the line graph of a tree in linear time: The maximal clique corresponds to the maximum degree in the original tree.
- *Lucky punch*: From a non-repetitive coloring of the edges, try to color the vertices of the line graph with the same colors but with the twist of putting a new unique color at some vertex. Try this for every possible vertex, if it works, the conjecture holds for this tree, else we no choice but to solve the LP.

6.4 Practical results for the small gap conjecture

Thanks to a list of all the non-isomorphic trees up to 22 vertices archived by Brendan McKay[4], we were able to test the small gap conjecture systematically against all trees up to 17 vertices. This represents 81'155 trees (16'661 caterpillars) and roughly two hours of computation. We also checked more than 40'000 trees on 18 vertices, which is already four hours of computation (models are increasingly difficult to solve and caterpillars become less commons). At this point we ran into memory issues related to `Python` and improving this would imply having better hardware or switching to a language which is more memory-efficient.

For all tree tested, the conjecture was verified with d at most 1: $\pi(L(T)) \leq \pi'(T) + 1$.

6.5 Getting the code

The whole codebase can be found on `GitHub` at the following address. Please take a look at the readme for technical details.

`github.com/DaiSijie/non-repetetive-graph-coloring`

[4] `http://users.cecs.anu.edu.au/~bdm/data/trees.html`

7 Conclusion

7.1 Summary of the results

In this project, we have:

- Shown how to apply the ideas of [?] to some other problems and extended it to edges.
- Obtained some advance on the small gap conjecture, notably lowered the gap between $\pi(L(T))$ and $\pi'(T)$ to $2\Delta(T)$ but also found other classes of graph for which the conjecture holds.
- Obtained a first bound for the Thue index in function of the treewidth and maximum degree, $\pi'(G) \in O(\Delta^2 t)$.
- Found a way to formulate our problem as an LP and a reduction to SAT
- Used it to build a codebase and verify the small gap conjecture for all tree up to 17 vertices. We have proposed a few optimizations to speed up the process, notably an algorithm to find separate paths lazily.

7.2 Future directions

Here are a few ideas for future research on this topic and things that we'd like to do:

- Improve the lower bound for the small gap conjecture. Notice that it will require a better understanding of lower bounds for $\pi'(T)$. For instance, is it possible to find an equivalent to Vizing's theorem for non-repetitive edge coloring? A straightforward improvement might be using the better k-special sequence described in [?] which uses only $3k + 1$ colors. As of this day, we were not able to find a counter-example. Using this other k-special sequence will require a different approach as it doesn't satisfy the order property lemma of S_k.
- Find a better edge levelling. This might yield a better bound than $\pi'(G) \in O(\Delta^2 t)$. For instance, we might require that each level is walk-non-repetitive.
- Try to optimize the memory usage of the program or use a cluster (the codebase is already fit for this use) and test the conjecture for more trees.