On $\lambda$-backbone coloring of cliques with tree backbones in linear time

Krzysztof Michalik$^1$  |  Krzysztof Turowski$^1$

$^1$Theoretical Computer Science Department, Jagiellonian University, Kraków, 30-348 Poland

Correspondence
Krzysztof Turowski, Theoretical Computer Science Department, Jagiellonian University, Krakow, 30-348 Poland
Email: krzysztof.szymon.turowski@gmail.com

Funding information
Krzysztof Turowski work was supported by Polish National Science Center 2020/39/D/ST6/00419 grant. For the purpose of Open Access, the author has applied a CC-BY public copyright license to any Author Accepted Manuscript (AAM) version arising from this submission.

A $\lambda$-backbone coloring of a graph $G$ with its subgraph (also called a backbone) $H$ is a function $c : V(G) \rightarrow \{1, \ldots, k\}$ ensuring that $c$ is a proper coloring of $G$ and for each $\{u, v\} \in E(H)$ it holds that $|c(u) - c(v)| \geq \lambda$.

In this paper we propose a way to color cliques with tree and forest backbones in linear time that the largest color does not exceed $\max\{n, 2\lambda\} + \Delta(H)^2 \lfloor \log n \rfloor$. This result improves on the previously existing approximation algorithms as it is $(\Delta(H)^2 \lfloor \log n \rfloor)$-absolutely approximate, i.e. with an additive error over the optimum.

We also present an infinite family of trees $T$ with $\Delta(T) = 3$ for which the coloring of cliques with backbones $T$ requires at least $\max\{n, 2\lambda\} + \Omega(\log n)$ colors for $\lambda$ close to $\frac{n}{2}$. The construction draws on the theory of Fibonacci numbers, particularly on Zeckendorf representations.

KEYWORDS
Graph coloring, backbone coloring, complete graphs, approximation algorithms, Fibonacci numbers

1 | INTRODUCTION

Within the general framework of graph coloring problems, there exists an important class of problems that is related to the frequency assignment problem: for a given set of transmitters (which are represented by the vertices of a graph) and their adjacency (i.e. adjacent transmitters are close enough or have a signal which is strong enough), assign the frequency bands to the transmitters in a way that keeps interference below a defined level while minimizing the total frequency span. In some
applications, it makes sense to distinguish a certain substructure of the network (called the backbone), designed as the most important part of communication, so it has to meet additional restrictions on the assignment.

This leads us to the backbone coloring problem, introduced by Broersma in [1]. First, let us define formally the backbone coloring, using a standard graph notation, e.g. from [2]:

**Definition** Let \( G \) be a graph and \( H \) be a subgraph of \( G \) with \( V(G) = V(H) \). Let also \( \lambda \in \mathbb{N}_+ \), \( \lambda \geq 2 \). The \( \lambda \)-backbone coloring of \( G \) with backbone \( H \) is defined as a function \( c : V(G) \to \mathbb{N}_+ \) such that

- \( c(u) \neq c(v) \) for every \( uv \in E(G) \),
- \( |c(u) - c(v)| \geq \lambda \) for every \( uv \in E(H) \).

Note that it differs from the vertex coloring problem in important ways: in an optimal \( \lambda \)-backbone coloring (i.e. using a minimum number of colors) the ordering of the colors matters, therefore we might observe that some smaller colors are not used while the larger ones are in use.

Broersma et. al. in [1] also defined the backbone coloring problem as an extension of a classical vertex coloring problem:

**Definition** Let \( G \) be a graph and \( H \) be a subgraph of \( G \) with \( V(G) = V(H) \). Let also \( \lambda \) be a natural number greater than 1. The \( \lambda \)-backbone coloring problem is defined as following: for a given positive integer \( k \), does there exist a \( \lambda \)-backbone coloring of \( G \) with backbone \( H \)?

The \( \lambda \)-backbone coloring number for a graph \( G \) with backbone \( H \) (denoted as \( \text{BBC}_\lambda(G, H) \)) is then defined as the smallest \( k \) such that there exists a \( \lambda \)-backbone coloring of \( G \) with backbone \( H \).

In general, it is straightforward to prove that the problem of determining \( \text{BBC}_\lambda(G, H) \) is \( \mathcal{NP} \)-hard [1]. In the same paper which introduced this problem there were shown basic bounds on the value of the \( \lambda \)-backbone coloring number, depending on \( \chi(G) \), the chromatic number of \( G \):

**Theorem 1 (Broersma et al., [1])** Let \( G \) be a graph and \( H \) its spanning subgraph. Then \( \chi(G) \leq \text{BBC}_\lambda(G, H) \leq \lambda(\chi(G) - 1) + 1 \).

In [3] and [4] there were proposed other bounds, suited particularly for graphs with \( \chi(H) \ll \chi(G) \):

**Theorem 2 (Havet et al., [3])** Let \( G \) be a graph and \( H \) its spanning subgraph. Then \( \text{BBC}_\lambda(G, H) \leq (\lambda + \chi(G) - 2)\chi(H) - \lambda + 2 \).

**Theorem 3 (Janczewski, Turowski, [4])** Let \( G \) be a graph on \( n \) vertices and \( H \) its spanning subgraph. Then \( \lambda(\chi(H) - 1) + 1 \leq \text{BBC}_\lambda(G, H) \leq \lambda(\chi(H) - 1) + n - \chi(H) + 1 \).

The \( \lambda \)-backbone coloring problem was studied for several classes of graphs, for example split graphs [5], planar graphs [3], complete graphs [6], and for several classes of backbones: matchings and disjoint stars [5], bipartite graphs [6] and forests [3]. For a special case \( \lambda = 2 \) it was also studied for many other cases e.g. triangle-free graphs with tree backbones [7], outerplanar graphs with matching backbones [8] and general graphs with bounded degree backbones [9] or with tree and path backbones [1].

In this paper, we turn our attention to the special case when the graph is complete (denoted \( K_n \)) and its backbone is a (nonempty) tree or a forest (which we will denote by \( T \) and \( F \), respectively). Note that it has a natural interpretation as a labeling problem: how to assign different labels to all vertices such that on every backbone edge the difference between labels is at least \( \lambda \). This description draws a comparison e.g. to \( L(k, 1) \)-labeling problem (see e.g. [10] for a survey), where the colors of any two adjacent vertices have to differ by at least \( k \) and the colors of any two vertices within distance 2 have to be distinct.
Broersma in [11] pointed out that the backbone coloring problem generalizes so called radio labeling problem: finding for a given graph $G$ the function $L: V(G) \rightarrow \mathcal{N}_+$ such that for every edge $uv \in E(G)$ it holds that $|L(u) - L(v)| \geq 2$ such that $RL(G) = \max_{v \in V(G)} L(v)$ is minimal – that is, it is exactly the case that $RL(G) = B\text{BC}_2(K_{|V(G)|}, G)$. However, note that the name radio labeling sometimes it refers to a different, although related problem [12, 13, 14].

In [15] it was proved that the radio labeling problem can be solved in polynomial time for graphs for which $k$-coloring can be found in polynomial time for some fixed $k$, e.g. for planar graphs or graphs with bounded treewidth. Additionally, [16] proved for comparability graphs we can find a partition of $V(G)$ into at most $k$ sets which induce semihamiltonian subgraphs in the complement of $G$ (i.e. it contains a Hamiltonian path) and from that it follows that $B\text{BC}_2(K_n, G)$ problem is solvable in polynomial time if $G$ is a comparability graph [11]. This generalizes an earlier result from [17], where (under the name of $L'(2, 1)$-labelings) it was shown that this problem is polynomially solvable on cographs, a subclass of comparability graphs.

On the hardness side, one has to point out that the radio labeling problem is equivalent to the travelling salesman problem with distances $B\text{BC}$ backbone coloring of $\Delta(G)$ back to the one in the previous section, as it shows that sometimes we need up to $\lambda > n$, and a result establishes additionally that depending on the choice of a backbone tree, the value of $B\text{BC}_2(K_n, T)$ can vary considerably, from $n$ to $\max\{n, 2\lambda\} + \Theta(\log n)$.

Finally, Section 4 concludes with a presentation of some open problems related to our work.
2 COMPLETE GRAPH WITH A TREE OR FOREST BACKBONE: AN ALGORITHM

In this section we will proceed as follows: we first introduce the so-called red-blue-yellow \((k, l)\)-decomposition of a forest \(F\) on \(n\) vertices, which finds a set \(Y\) of size at most \(l\) such that we can split \(V(F) \setminus Y\) into two independent and almost equal sets \(R, B\) (with \(|R| - |B| \leq k\)). Then, we show that we can color \(R\) and \(B\) with sets of consecutive colors and assign to vertices from \(Y\) the smallest and the largest colors in a way that in total we find a \(\lambda\)-backbone coloring in which the maximum color does not exceed \(\max\{n, 2\lambda\}\) more than \(\Delta^2(F) \log n\).

We start with a few remarks on notation and some definitions. Let \(c\) be a unique 2-coloring (up to the permutations of colors) of a tree \(T\). Let us also define \(C_i(T) = \{v \in V(T) : \ c(v) = i\}\), i.e. the number of vertices in color \(i\), for \(i = 1, 2\). Assume without loss of generality that \(c\) is such that \(|C_1(T)| \geq |C_2(T)|\).

Throughout the paper we will use the concept of tree imbalance, defined formally as:

**Definition** Let \(T\) be a tree and \(c\) its unique 2-coloring with \(|C_1(T)| \geq |C_2(T)|\). Let the imbalance number of \(T\) be \(\text{imb}(T) := |C_1(T)| - |C_2(T)|\).

From this definition, it directly follows that \(\text{imb}(T) \geq 0\).

First, let us prove the simple structural fact about trees in general:

**Lemma 1** In every tree \(T\) there exists a central vertex \(v \in V(T)\) such that every connected component of \(T - v\) has at most \(|V(T)|/2\) vertices.

**Proof** The idea is to start from any vertex \(u\), and then jump to its neighbor with the largest component size in \(T - u\), until we hit a vertex with desired property. Note that for any vertex \(v\) there can be at most one neighbor \(u\) such that its connected component \(T_u\) in \(T - v\) has more than \(|V(T)|/2\) vertices, so the jumps are unique.

Suppose that the number of vertices in \(T_u\) is equal to \(k\). Then the number of vertices in all components of \(T - v\) sum to \(|V(T)| - k - 1\). Now if we look at \(T - u\), then it has
- one connected component containing \(v\) with \((|V(T)| - k - 1) + 1 < |V(T)|/2\) vertices,
- all other components which are subtrees of \(T_u - u\), so they contain at most \(k - 1\) vertices.

Thus, every jump reduces the size of the largest connected component by at least 1 – and therefore the algorithm always terminates correctly.

**Definition** Let \(T\) be a tree on \(n\) vertices. We call a partition \((R, B, Y)\) of \(V(T)\) a red-blue-yellow \((k, l)\)-decomposition of \(T\) if \(|Y| \leq l\) and \(R\) and \(B\) are independent sets with \(|R| - |B| = k\).

From this definition it follows that if \((R, B, Y)\) is a red-blue-yellow \((k, l)\)-decomposition, then \((B, R, Y)\) is a red-blue-yellow \((-k, l)\)-decomposition.

**Lemma 2** If \(T\) is a tree with a root \(v \in V(T)\) and \(T_1, \ldots, T_k\) are subtrees of \(T\), made by removing \(v\) from \(T\), then \(\text{imb}(T) \leq 1 + \sum_{i=1}^{k} \text{imb}(T_i)\).

**Proof** Let us denote by \(v_i\) the root of \(T_i\) for \(i = 1, \ldots, k\). Observe that
- either \(C_i(T_i) \subseteq C_1(T)\) and \(C_2(T_i) \subseteq C_2(T)\) (when \(v_i \in C_2(T)\)),
- or \(C_1(T_i) \subseteq C_2(T)\) and \(C_2(T_i) \subseteq C_1(T)\) (when \(v_i \in C_1(T)\)).
Thus, the imbalance of $T$ can be expressed in the following formula (where 1 appears because of the vertex $v$):

$$imb(T) = |C_1(T)| - |C_2(T)| \leq 1 + \sum_{j=1}^{k} (|C_1(T_j)| - |C_2(T_j)|) = 1 + \sum_{j=1}^{k} imb(T_j).$$

**Lemma 3** For any tree $T$ on $n$ vertices and for any $k \in [0, imb(T)]$ there exists a red-blue-yellow $(k, \lceil \log n \rceil)$-decomposition of $T$.

**Proof** We prove existence of such a decomposition by constructing it.

We begin by assigning $R_0, B_0, Y_0, T_0$ as initial sets of $R, B, Y, T$ and for convenience, we initialize $R_0$ with a set $D$ of $k$ dummy isolated vertices, while $B_0 = Y_0 = \emptyset$ and $T_0 = V(T)$. Note, that any red-blue-yellow $(0, \lceil \log n \rceil)$-decomposition of $T \cup D$ can be directly turned into a red-blue-yellow $(k, \lceil \log n \rceil)$-decomposition of $T$ just by removing vertices from $D$ and possibly swapping first two sets.

That is why, from now on, we will concentrate on balancing both of the sets, so that they end up with the same amount of vertices. For that purpose, we will be constructing iteratively sets $R_i, B_i, Y_i, T_i$ and introduce invariants that will hold at each step while decomposing $V(T)$ into $R, B, Y$, which guarantee the convergence to a desired solution:

1. $V(T) \cup D = R \cup B \cup Y \cup V(T_i)$,
2. $T_i$ is a tree,
3. $R_i$ and $B_i$ are independent sets,
4. $T_i$ does not have neighbors in $R_i \cup B_i$,
5. $0 \leq |R_i| - |B_i| \leq imb(T_i)$.

Of course, all of the above conditions hold for $i = 0$.

To construct $(R_{i+1}, B_{i+1}, Y_{i+1}, T_{i+1})$ from $(R_i, B_i, Y_i, T_i)$, we do as follows:

Consider vertex $v$ that we find using Lemma 1 in $T_i$ and denote by $T'_j$ ($j = 1, \ldots, d$) the subtrees obtained by removing $v$ from $T_i$. Let us consider two cases, depending on which of the values $|R_i| - |B_i|$ and $\sum_{j=1}^{d} imb(T'_j)$ is larger.

Case $A$: if it holds that $|R_i| - |B_i| > \sum_{j=1}^{d} imb(T'_j)$. However from invariants (1) and (2) we also know that $imb(T_i) \geq |R_i| - |B_i|$. Therefore $imb(T_i) > \sum_{j=1}^{d} imb(T'_j)$, but from Lemma 2 we know that $imb(T_i) \leq 1 + \sum_{j=1}^{d} imb(T'_j)$, so it has to be the case that

$$1 + \sum_{j=1}^{d} imb(T'_j) = imb(T_i) = |R_i| - |B_i|.$$ Then, we observe that for $R = R_i \cup C_2(T_i), B = B_i \cup C_1(T_i), Y = Y_i$ it is true that:

- $(R, B, Y)$ is a partition of $V(T) \cup D$,
- $R$ and $B$ are both independent sets – due to the invariants (1) and (2),
- $|R| - |B| = (|R_i| - |B_i|) - (|C_1(T_i)| - |C_2(T_i)|) = imb(T_i) - imb(T_i) = 0$,
- $|Y_i| = i \leq \lceil \log_2 n \rceil$.

Therefore, $(R, B, Y)$ is a red-blue-yellow $(0, \lceil \log n \rceil)$-decomposition of $T \cup D$.

Case $B$: if it is true that $|R_i| - |B_i| \leq \sum_{j=1}^{d} imb(T'_j)$.

Until now we never needed to order subtrees in any way, so we can safely assume that they are sorted nondecreasingly according to their imbalance (i.e. $imb(T'_1) \leq \cdots \leq imb(T'_d)$). We will be iterating through $T'_i$ and put them in either of the two sets, $R$ or $B$.

Let us define intermediate sets $R'_j, B'_j$ as sets that we get after we processed first $j$ subtrees. Clearly $R'_0 = R_i$ and $B'_0 = B_i$.

To get next pair of sets $R'^{i+1}_j, B'^{i+1}_j$ we will follow a very simple rule - we choose a smaller set (say it is $B'_j$) and add $C_1(T'_j)$ to it, while we add $C_2(T'_j)$ to $R'_j$. If sets are equal, we choose one of them arbitrarily. If necessary, we swap sets, so $|R'^{i+1}_j| > |B'^{i+1}_j|$ holds after each iteration. We iterate this procedure for $j = 1, \ldots, d - 1$, while the final tree, $T'^d_i$ becomes our $T_{i+1}$, the final sets $R'^{d-1}_j$ and $B'^{d-1}_j$ become our $R_{i+1}$ and $B_{i+1}$, and we proceed to the next iteration.
Let us now prove that the invariants (1)–(5) are preserved between iterations. The invariant (1) holds directly by the construction of the respective sets:

- we assign \( u_i \) to \( Y_{i+1} \),
- we distribute all vertices of subtrees \( T_i^j \) for \( j = 1, \ldots, d-1 \) between \( R_{i+1} \) and \( B_{i+1} \),
- the remaining subtree \( T_i^d \) becomes the new \( T_{i+1} \).

Moreover, the invariant (2) is trivially preserved directly by the fact that \( T_{i+1} \) is a subtree of \( T_i \).

The invariant (3) states that \( R_i \) and \( B_i \) are independent sets. From the invariant (4) it follows that if we add to them any independent sets from \( T_j \), they still remain independent. Thus, since \( C_1(T_i^j) \) and \( C_2(T_i^j) \) are independent, and there is no edge between \( C_i(T_i^j) \) and \( C_i(T_i^{j'}) \) for any \( l = 1, 2 \) and \( j \neq j' \), it follows that \( R_{i+1} \) and \( B_{i+1} \) are also independent sets – and therefore the invariant (3) is true also for \( i+1 \).

By the invariant (4) for \( i \), we know that \( T_i \) has its neighbors only in \( Y_i \). Thus, \( T_{i+1} \), is not only a subtree of \( T_i \), but it is only adjacent to \( Y_i \) or \( v \) itself, that is, only to \( Y_{i+1} \) – thus the invariant (2) holds for \( i+1 \).

The crucial observation to prove the invariant (5) is that after \( j \)-th iteration of the inner for loop we have

\[
0 \leq |R_i^j| - |B_i^j| \leq \sum_{j'=j+1}^{d} \text{imb}(T_i^{j'}).
\]

Clearly this holds for \( j = 0 \) as this condition is identical to \( 0 \leq |R_0| - |B_0| \leq \sum_{j=1}^{d} \text{imb}(T_i^j) \). To prove it inductively for \( j = 1, 2, \ldots, d-1 \), we distinguish two cases:

- if \( |R_i^{j-1}| - |B_i^{j-1}| < \text{imb}(T_i^j) \), then

\[
|R_i^j| - |B_i^j| = |R_i^{j-1}| + |C_2(T_i^j)| - |B_i^{j-1}| - |C_1(T_i^j)| = \left| |C_2(T_i^j)| - \text{imb}(T_i^j) \right| \\
\leq \max\{|R_i^{j-1}| - |B_i^{j-1}|, \text{imb}(T_i^j)\} = \text{imb}(T_i^d),
\]

- if \( |R_i^{j-1}| - |B_i^{j-1}| \geq \text{imb}(T_i^d) \), then

\[
|R_i^j| - |B_i^j| = |R_i^{j-1}| + |C_2(T_i^j)| - |B_i^{j-1}| - |C_1(T_i^j)| = |R_i^{j-1}| - |B_i^{j-1}| - \text{imb}(T_i^j) \\
\leq \sum_{j'=j}^{d} \text{imb}(T_i^{j'}) - \text{imb}(T_i^j) = \sum_{j'=j+1}^{d} \text{imb}(T_i^{j'}),
\]

where the second to last inequality follows from the induction assumption that \( |R_i^{j-1}| - |B_i^{j-1}| \leq \sum_{j'=j}^{d} \text{imb}(T_i^{j'}) \).

Ultimately, for \( j = d-1 \) we get in either case that \( |R_{i+1}| - |B_{i+1}| = |R_i^{d-1}| - |B_i^{d-1}| \leq \text{imb}(T_i^d) = \text{imb}(T_{i+1}) \), and clearly by construction it is always true that \( |R_i^j| - |B_i^j| \geq 0 \) so the invariant (5) is also preserved for \( i+1 \).

The above inductive proof establishes that all the invariants are true at the beginning of the iteration when \( V(T_k) = \emptyset \). Therefore, from invariant (1) it follows that \((R_k, B_k, Y_k)\) is a partition of \( V(T) \cup D \). From invariants (1) and (3) it follows that \( 0 \leq |R_k| - |B_k| \leq \text{imb}(T_k) = 0 \) and that \( R_k \) and \( B_k \) are independent sets, respectively. Finally, from Lemma 1 we know that \( \frac{|V(T_i)|}{2} \leq \frac{|V(T_{i+1})|}{2} \) for all \( i = 0, 1, \ldots, \) so \( k \leq |\log n| \) and therefore \( |Y_k| \leq |\log n| \) Thus, \((R_k, B_k, Y_k)\) is a red-blue-yellow \((0, |\log n|)\)-decomposition of \( T \cup D \).

Note that the proof above can be directly translated to Algorithm 1. In addition,

**Theorem 4** Algorithm 1 with an appropriate preprocessing runs in \( O(n) \) time.
Algorithm 1 Find a red-blue-yellow \((k, \lceil \log n \rceil\))-decomposition of \(T\).

1: \textbf{function} \textsc{RBY-decompose}(\(T, k\))
2: \hspace{1em} \(D \leftarrow\) a set of dummy \(k\) isolated vertices
3: \hspace{1em} \(T_0 \leftarrow T, R_0 \leftarrow D, B_0 \leftarrow \emptyset, Y_0 \leftarrow \emptyset\)
4: \hspace{1em} \textbf{for} \(i = 0, 1, 2, \ldots\) \textbf{do}
5: \hspace{2em} \textbf{if} \(V(T_i) == 0\) \textbf{then}
6: \hspace{3em} \(R \leftarrow R_i, B \leftarrow B_i, Y \leftarrow Y_i\)
7: \hspace{3em} \textbf{break}
8: \hspace{2em} \textbf{end if}
9: \hspace{1em} \textbf{end for}
10: \hspace{1em} \begin{align*}
& T'_1, \ldots, T'_d \leftarrow \text{subtrees of } T_i \text{ rooted in } v \text{ with nondecreasing } \text{imb}(T'_i) \\
& \textbf{if} |R_i| - |B_i| > \sum_{j=1}^{d} \text{imb}(T'_j) \text{ then}
& \hspace{1em} R \leftarrow R_i \cup C_2(T_i), B \leftarrow B_i \cup C_1(T_i), Y \leftarrow Y_i
& \hspace{1em} \textbf{break}
& \textbf{end if}
11: \hspace{1em} \textbf{else}
12: \hspace{2em} R_i(0) \leftarrow R_i, B_i(0) \leftarrow B_i
13: \hspace{2em} \textbf{for} \(j = 1, 2, \ldots, d - 1\) \textbf{do}
14: \hspace{3em} R_i(j) \leftarrow R_i(j - 1) \cup C_2(T'_j), B_i(j) \leftarrow B_i(j - 1) \cup C_1(T'_j)
15: \hspace{3em} \textbf{if} |R_i(j)| < |B_i(j)| \text{ then}
16: \hspace{4em} \text{Swap } R_i(j) \text{ and } B_i(j)
17: \hspace{3em} \text{end if}
18: \hspace{2em} \textbf{end for}
19: \hspace{1em} R_{i+1} \leftarrow R_i(d - 1), B_{i+1} \leftarrow B_i(d - 1), Y_{i+1} \leftarrow Y_i \cup \{v_i\}, T_{i+1} \leftarrow T'_d
20: \hspace{1em} \textbf{if} |R \setminus D| < |B \setminus D| \text{ then}
21: \hspace{2em} \text{Swap } R \text{ and } B
22: \hspace{1em} \textbf{end if}
23: \hspace{1em} \textbf{return} \((R \setminus D, B \setminus D, Y)\)

Proof Observe that for a tree on \(n\) vertices we can compute for every vertex \(v\) and its neighbor \(u\) functions \(f(v, u)\) and \(g(v, u)\) denoting the sizes of subsets of \(C_1(T)\) and \(C_2(T)\) restricted to the connected component containing \(u\) in \(T - v\). Moreover, it can be done in linear time: it is sufficient to root \(T\) in an arbitrary vertex, compute values of all \(f(v, u)\) and \(g(v, u)\) when \(u\) is a child of \(v\) recursively, and then by another recursion get the missing values of \(f(v, \text{parent}(v))\) and \(g(v, \text{parent}(v))\). Note that this way we can compute both size of the component (equal to \(f(v, u) + g(v, u)\)) as well as its imbalance (equal to \(f(v, u) - g(v, u)\)) on request in constant time.

Next, let us count the total number of jumps necessary for finding central vertices over all loops in Algorithm\[1\] As it was stated in the proof of Lemma\[1\], while searching for a central vertex we always jump from a vertex to its neighbor in a way that decreases the largest remaining component by one. Thus, if in the next iteration we start at exactly the neighbor of the previous central vertex, there can be only \(O(n)\) such jumps in total.

Additionally, each jump can be made in constant time, provided that we sorted all neighbors of \(v\) by their subtree sizes, that is, the values of \(f(v, u) + g(v, u)\) – and that we can do also at the beginning using e.g. bucket sorting in \(\deg(v)\) time per vertex (so in \(O(n)\) time for all the vertices). Later, it is then sufficient to update the sizes lazily: only when we jump to certain \(v\) from \(u\), we fix its values \(f(v, u)\) and \(g(v, u)\), and move it to the proper bucket in the sorted sequence.

Note that the rest of the \(i\)-th iteration of the main loop in the algorithm takes time proportional to

\[
\deg(v_i) + |R_{i+1} \cup B_{i+1} \cup Y_{i+1}| - |R_i \cup B_i \cup Y_i| + O(1),
\]
so by telescoping sum over all (at most $[\log n]$) iterations we directly obtain that the total time is also clearly bounded by
\[\sum \deg(v) + |R \cup B \cup Y| + O(\log n) \leq 3|V(T)| + O(\log n) = O(n).\]

For completeness, we note that Lemma 4 can be extended to forests:

**Corollary 1** For a forest $F = \bigcup_{i=1}^{r} T_i$ on $n$ vertices consisting of $r$ trees and any $k \in [0, \sum_{i=1}^{r} \imb(T_i)]$ there exists a red-blue-yellow $(k, [\log n])$-decomposition of $F$.

In order to formulate our results for the $\lambda$-backbone coloring problem, we need a simple lemma, which enables us to color a clique with a forest backbone with a given set of colors:

**Lemma 4** Let $F$ be a forest on $n$ vertices with a partition $V(F) = A \cup B$ into disjoint independent sets $A$ and $B$. Let also $[a_1, a_2]$ and $[b_1, b_2]$ be intervals such that $a_2 - a_1 \geq |A| - 1$, $b_2 - b_1 \geq |B| - 1$, $a_1 + \lambda \leq b_1$ and $a_2 + \lambda \leq b_2$.

Then, there exists a $\lambda$-backbone coloring $c$ of $K_n$ with backbone $F$ such that $c(v) \in [a_1, a_2]$ for every $v \in A$ and $c(v) \in [b_1, b_2]$ for every $v \in B$. Moreover, this coloring can be found in $O(n)$ time.

**Proof** If $F$ contains only isolated vertices, then we can assign in any order colors from $[a_1, a_2]$ to $A$ and from $[b_1, b_2]$ to $B$, as it clearly would be a $\lambda$-backbone coloring with the required properties.

If $F$ has a leaf $v \in B$ with a neighbor $u \in A$, then we can assign $c(v) = b_1$, $c(u) = a_1$ and invoke a subproblem for $F' = F - \{u,v\}$, $A' = A \setminus \{u\}$, $B' = B \setminus \{v\}$ with the same coloring $c$ and color intervals $[a_1 + 1, a_2]$ and $[b_1 + 1, b_2]$. The solution for $F'$ would be consistent with coloring of $u$ and $v$, as all other neighbors of $u$ in $F$ would get colors at least $b_1 + 1 \geq \lambda + a_1 + 1 > \lambda + c(u)$.

Otherwise, $F$ has a leaf $v \in A$ with a neighbor $u \in B$. We can assign $c(v) = a_2$, $c(u) = b_2$ and invoke a subproblem for $F' = F - \{u,v\}$, $A' = A \setminus \{v\}$, $B' = B \setminus \{u\}$ with the same coloring $c$ and color intervals $[a_1, a_2 - 1]$ and $[b_1, b_2 - 1]$. The solution for $F'$ would be consistent with coloring of $u$ and $v$, since all other neighbors of $u$ in $F$ would get colors at most $a_2 - 1 \leq b_2 - 1 - \lambda < c(u) - \lambda$.

The linear running time follows directly from the fact that we compute $c$ only once and we can pass additionally through recursion the lists of leaves and isolated vertices in an uncolored induced subtree. The total number of updates of these lists is proportional to the total number of edges in the tree, hence the claim follows.

The lemma above can be applied to the $\lambda$-backbone coloring problem directly. Note that here we can extend the notation $C_1$, $C_2$ used before only for trees – however this time it is relative to a 2-coloring $c$, which in the case of forests may not be unique:

**Theorem 5** Let $F$ be a forest on $n$ vertices and $c$ be a 2-coloring of $F$ such that $|C_1(F)| \geq |C_2(F)|$, where $C_i(F) = \{v \in V(F) : c(v) = i\}$. It holds that $\BBC_\lambda(K_n, F) \leq \max\{\lambda + |C_1(F)|, n\}$. Moreover, we can find a respective coloring in $O(n)$ time.

**Proof** Let $L = \max\{\lambda, C_2(F)\}$. Let also $A = C_2(F)$, $B = C_1(F)$ with $[a_1, a_2] = [1, |C_2(F)|]$ and $[b_1, b_2] = [L + 1, L + |C_1(F)|]$. These choices meet the conditions of Lemma 4 since $a_2 - a_1 = |A| - 1$, $b_2 - b_1 = |B| - 1$, $a_1 + \lambda \leq a_1 + L = b_1$, and $a_2 + \lambda \leq |C_2(F)| + L = |C_1(F)| + L = b_2$.

Therefore, we directly infer that there exists a $\lambda$-backbone coloring $c$ using only colors no greater than $L + |C_1(F)|$.

However, we can also couple Lemma 4 with a red-blue-yellow decomposition to obtain a different bound:

**Theorem 6** For a forest $F$ on $n$ vertices it holds that $\BBC_\lambda(K_n, F) \leq \max\{n, 2\lambda + \Delta^2(F)[\log n]\}$. Moreover, we can find a respective coloring in $O(n)$ time.
Proof Let $L = \max \left\{ \frac{1}{2}, \lambda \right\}$. From Corollary 1, we can get a red-blue-yellow ($0, \lceil \log n \rceil$)-decomposition ($R, B, Y$) of $F$. Note that by construction it holds that $|R| = |B| \leq L$.

Observe that $Y$ is not necessarily an independent set. Therefore, we need to guarantee the correctness of the coloring for the edges with both endpoints in $Y$. We handle that in the following way: we obtain $F'$ from $F$ by contracting all edges with both endpoints in $R \cup B$. From an arbitrary 2-coloring $c'$ of $F'$ we construct a partition of $Y = Y_1 \cup Y_2$ such that both $Y_i = \{ v \in V : c'(v) = i \}$. Without loss of generality assume that $|Y_1| \geq |Y_2|$.

Let $\mathcal{N}_G(U)$ be a set of neighbors of vertices from $U$ in $G$, i.e.

$$\mathcal{N}_G(U) = \{ v \in V(G) : \exists_{u \in U} \{ u, v \} \in E(G) \}$$

Let us introduce $B_1 = B \cap \mathcal{N}_F(Y_1)$ and $R_1 = R \cap \mathcal{N}_F(B_1)$. Let also $R_2 = R \cap \mathcal{N}_F(Y_2)$, $B_2 = B \cap \mathcal{N}_F(R_2)$ and $B^* = B \setminus (B_1 \cup B_2)$, $R^* = R \setminus (R_1 \cup R_2)$. Note that from these definitions it follows that $B_1 \cap B_2 = R_1 \cap R_2 = \emptyset$. If this was not the case, then it would mean that there exists a path $(u_1, v_1, v_2, u_2)$ in $F$ such that $u_1 \in Y_1$, $u_2 \in Y_2$ and $v_1, v_2 \in R \cup B$. However, this would mean that in $F'$ both $u_1$ and $u_2$ had to be assigned the same color – so they could not be in $Y_1$ and $Y_2$, respectively.

To simplify the notation we define $D = \Delta(F)$. We will proceed with the coloring as follows:

- vertices from $Y_1$ will get colors from 1 to $|Y_1|$,
- vertices from $B$ will get colors from $D|Y_1| + 1$ to $L + D|Y_1|$,
- vertices from $R$ will get colors from $L + D^2|Y_1| + 1$ to $2L + D^2|Y_1|$, 
- vertices from $Y_2$ will get colors from $2L + D^2|Y_1| + (D - 1)|Y_1| + 1$ to $2L + D^2|Y_1| + D|Y_2|$.

We will color $F$ by assigning colors to $Y_1$, $B_1$ and $R_1$ first, and then to $Y_2$, $R_2$ and $B_2$, symmetrically. Finally, we prove that we are left with the part of the set $R^* \cup B^*$ that can be colored using the remaining colors (see Figure 4 for the assignment of colors to the sets of vertices).

First, let us arrange vertices from $Y_1$ in any order and assign them colors from 1 to $|Y_1|$. Now, let us take vertices from $B_1$ and assign them the largest colors from the set $\{D|Y_1| + 1, L + D|Y_1|\}$ in the following way: the vertex with the largest colored neighbor in $Y_1$ gets $L + D|Y_1|$, second largest gets $L + D|Y_1| - 1$, etc. (ties are broken arbitrarily). Such a coloring ensures two things: first, all color constraints between $Y_1$ and $B_1$ are met, and second, the smallest color used for $B_1$ is at least $L + 1$.

Similarly, we color $R_1$: we sort the vertices according to the largest colors of their neighbors in $B_1$ and assign the largest colors to the ones with the largest neighbor color. Thus we are certain that all color constraints between $B_1$ and $R_1$ are met and that the smallest color used for $R_1$ is at least $2L + D|Y_1| + 1$, since every vertex from $B_1$ has at most $D - 1$ neighbors in $R_1$.

Symmetrically, in the same fashion, we can color $Y_2$, $R_2$, and $B_2$, only this time assigning the smallest colors available for each set (see Figure 4).

Since there are at most $L$ vertices in both $R$ and $B$, we are sure that the sets of colors assigned to $B_1$ and $B_2$ (or $R_1$ and $R_2$, respectively) do not overlap. Furthermore, by construction, the only remaining vertices to be colored are exactly the ones from the set $R^* \cup B^*$. Note that any $v \in B^*$ can be adjacent only to some $u \in Y_2 \cup R_1$ (or to $u \in R^*$) – but the largest color to be used for $B^*$ (certainly at most the largest color for $B$, i.e. $L + D|Y_1|$) is at least lower by $L$ from the smallest color used in $Y_2 \cup R_1$ (i.e. $2L + D^2|Y_1| - D(D - 1)|Y_1| = 2L + D|Y_1|$). Again, the symmetrical argument goes for $R^*$.

By the argument above, the remaining colors for $B^*$ and $R^*$ form intervals $[b_1, b_2]$ and $[r_1, r_2]$ such that $b_1 + L \leq r_1$, $b_2 + L \leq r_2$. Thus, we are in position to color $F[R^* \cup B^*]$ with its disjoint independent sets $R^*$, $B^*$ and colors $[b_1, b_2]$, $[r_1, r_2]$ using Lemma 2.

To obtain the total running time we first note that each of the initial steps – obtaining $(R, B, Y)$ from Corollary 1 (e.g. using Algorithm 1), contraction of $F$ into $F'$, and finding both $Y_1$ and $Y_2$ – requires only linear time. Coloring $Y_1 \cup R_1 \cup B_1$ also requires $O(n)$ time, since we need to traverse each edge between these vertices only once to ensure the proper distances between the colors, and it is sufficient to use bucket sort to order vertices within $B_1$ and $R_1$. The same argument follows symmetrically for
Finally, Theorem 5 guarantees that the coloring of the remaining vertices from $R^* \cup B^*$ also can be found in $O(n)$ time.

Figure 1 The main idea for the $\lambda$-backbone coloring based on a red-blue-yellow decomposition. For brevity, $M = 2L + D^2|Y_1| + D|Y_2|$ denotes the maximum color used.

$Y_2 \cup R_2 \cup B_2$. Finally, Theorem 5 guarantees that the coloring of the remaining vertices from $R^* \cup B^*$ also can be found in $O(n)$ time.

**Theorem 7** There exists an algorithm with running time $O(n)$ which for any forest $F$ on $n$ vertices with $\Delta(F) \geq 2$ finds a $\lambda$-backbone coloring such that $B\overline{B}C_\lambda(K_n, F) \leq \text{OPT} + \Delta^2(F)[\log n]$.

**Proof** First, we prove the theorem for tree backbones. Let us run the algorithms from Theorem 5 and Theorem 6 and return the better result. This ensures that

\[
B\overline{B}C_\lambda(K_n, T) \leq \min\{\max\{\lambda + |C_1(T)|, n\}, \max\{n, 2\lambda + \Delta^2(T)[\log n]\}\}
\leq \max\{\min\{\lambda + |C_1(T)|, 2\lambda\}, n\} + \Delta^2(T)[\log n].
\]

Let us also observe that if $\Delta(T) \geq 2$ (that is, if $T$ contains at least one edge) for the optimal $\lambda$-backbone coloring $c$ for a graph $K_n$ with backbone $T$ it holds that

- either $\frac{1}{2} \max_{e \in V(T)} c(e) = 2$, so $\left\lceil \frac{c}{2} \right\rceil$ is a 2-coloring of $T$ – then it is clear that $\max_{e \in V(T)} c(e) \geq \lambda + |C_1(T)|$. 


We obtained a contradiction. Thus, either
\[ \frac{1}{\lambda} \max_{v \in V(T)} c(v) > 2, \text{ but } [\frac{1}{\lambda}] \text{ has to be a valid coloring of } T, \text{ but it has to use more than 2 colors – so } \max_{v \in V(T)} c(v) \geq 2\lambda + 1. \]

Therefore, it is true that \( B\overline{B}C_{\lambda}(K_n, T) \geq \min\{\lambda + |C_1(T)|, 2\lambda + 1\} \). And of course we have to use a different color for each vertex, so \( B\overline{B}C_{\lambda}(K_n, T) \geq n \) – thus \( B\overline{B}C_{\lambda}(K_n, T) \geq \max\{\min\{\lambda + |C_1(T)|, 2\lambda + 1\}, n\} \). Combining all these bounds we obtain the desired result.

To achieve the same result for forest backbones we only need to add some edges that would make the backbone connected and spanning. However, we can always make a forest connected by adding edges between some leaves and isolated vertices and we will not increase the maximum degree of the forest, as long as \( \Delta(F) \geq 2 \).

\section{Complete Graph with a Tree or Forest Backbone: A Lower Bound}

In this section, we prove that there exists a family of trees with the maximum degree 3 for which \( B\overline{B}C_{\lambda}(K_n, T) \geq \max\{n, 2\lambda\} + \Theta(\log n) \) in four steps. First, we will define the family of trees with parameters directly related to the Fibonacci numbers.

Next, we will show that the existence of a red-blue-yellow \((k, l)\)-decomposition implies the existence of another red-blue-yellow \((k', l)\)-decomposition with an additional property. Then, we will prove that any such decomposition would imply further that there holds a certain decomposition of some large number (i.e. half of the large Fibonacci number) into a sum and a difference of a small number of Fibonacci numbers. Finally, we will establish that such a decomposition cannot exist – therefore there is no red-blue-yellow \((k, l)\)-decomposition for our trees assuming that \( k \) and \( l \) are proportional to \( \log n \).

We begin by connecting our red-blue-yellow \((k, l)\)-decomposition to the \( \lambda \)-backbone coloring problem:

**Theorem 8** Let \( T \) be a tree on \( n \) vertices. Let also \( \lambda \geq 2 \text{ and } l \) be any positive integer such that \( 2\lambda + l \geq n \). If \( B\overline{B}C_{\lambda}(K_n, T) \leq 2\lambda + l \), then \( T \) has a red-blue-yellow \((k, l)\)-decomposition for some \( k \in [0, \min\{\lambda - 1, 2\lambda - n + l\}] \).

**Proof** Suppose that \( T \) is a tree that does not have any red-blue-yellow decomposition for any \( k \in [0, \min\{\lambda - 1, 2\lambda - n + l\}] \), but at the same time it is true that \( B\overline{B}C_{\lambda}(K_n, T) \leq 2\lambda + l \) with some optimal \( \lambda \)-backbone coloring \( c \).

Now we can define the following sets:

- \( R = \{ v \in V(T) : 1 \leq c(v) \leq \lambda \} \),
- \( B = \{ v \in V(T) : \lambda + 1 \leq c(v) \leq 2\lambda \} \),
- \( Y = \{ v \in V(T) : c(v) > 2\lambda \} \).

Since all vertices in \( c \) have different colors, it is true that \( |Y| \leq l \). Moreover, the optimality of \( c \) implies that both \( R \) and \( B \) are non-empty. From the fact that \( c \) is a coloring of \( K_n \) it follows that both \( R \) and \( B \) contain at most \( \lambda \) vertices since all the vertices have different colors. And by the definition of the backbone coloring, \( R \) and \( B \) have to be independent sets in \( T \).

Clearly, it holds that \( k' = |R| + |B| \leq \lambda - l \). Moreover, since by a counting argument both \( R \) and \( B \) have to contain at least \( n - \lambda - |Y| \) vertices (otherwise the other one would have more than \( \lambda \) vertices), we know that \( k' \leq \lambda - (n - \lambda - |Y|) \leq 2\lambda - n + l \).

Thus, either \((R, B, Y)\) or \((B, R, Y)\) is a red-blue-yellow \((k', l)\)-decomposition of \( T \) with \( k' \in [0, \min\{\lambda - 1, 2\lambda - n + l\}] \)—and we obtained a contradiction.

Now we define a family of Fibonacci trees, built recursively:

**Definition** We call a rooted tree the \( N \)-th Fibonacci tree (denoted as \( T^F_N \)) if:

- \( T^F_1 = T^F_2 = K_1 \),
- \( T^F_N \) for \( N \geq 3 \) is a tree with root \( u_1 \) with only child \( u_2 \) such that \( u_2 \) has two subtrees, which are, respectively, \((N - 1)\)-th and \((N - 2)\)-th Fibonacci trees.
We will ultimately show that this class contains infinitely many trees (starting from certain $N_0$), presented in Figure 2 cannot be colored using less than $\max\{n, 2\lambda\} + \frac{1}{48} \log_\phi n - 3$ colors with $\phi = \frac{1+\sqrt{5}}{2}$.

Here we will also make use of $C_1(T^F_N)$ and $C_2(T^F_N)$ sets. Without a loss of generality from now on we always define these sets with respect to a 2-coloring $c$ of $T^F_N$ where the root of $T^F_N$ is colored with 1.

**Proposition 1** For every $N$ each vertex of $T^F_N$ is either the root or the parent of the root of some $T^F_i$ subtree for some $i \in [1, N]$.

**Theorem 9** If $T^F_N$ is the $N$-th Fibonacci tree, then $|V(T^F_N)| = 3F_N - 2$, where $F_N$ is the $N$-th Fibonacci number.

Moreover, for $n \geq 10$ it holds that $N = \lfloor \log_\phi n \rfloor$ (with $\phi = \frac{1+\sqrt{5}}{2}$).

**Proof** The first part of the theorem obviously holds for $N = 1, 2$. If we assume that the theorem holds for all $i \in [1, N-1]$ for some $N \geq 3$, then

$$|V(T^F_N)| = |V(T^F_{N-1})| + |V(T^F_{N-2})| + 2 = 3F_{N-1} + 3F_{N-2} - 2 + 2 = 3F_N - 2.$$ 

The second part of the theorem can be proved in the following way: $n = 3F_N - 2$ implies that $\frac{1}{3} \phi^n (n + 2) = \phi^n - (-\phi)^{-N}$. Therefore, we know that for $n \geq 2$ ($N \geq 3$) it holds that

$$\log_\phi (\phi^n - (-\phi)^{-N}) \geq N + \log_\phi (1 - \phi^{-2N}) \geq N + \frac{\phi^{-2N}}{\ln \phi} - \frac{\phi^{-2N}}{\ln \phi} > N - \frac{1}{5},$$

where we used the fact that $\ln(1 - x) \geq -x - x^2$ for $x \in (0, 0.68)$.

On the other hand,

$$\log_\phi (\phi^n - (-\phi)^{-N}) \leq N + \log_\phi (1 + \phi^{-2N}) \leq N + \frac{\phi^{-2N}}{\ln \phi} + \frac{\phi^{-2N}}{\ln \phi} < N + \frac{1}{5},$$

where we used the fact that $\ln(1 + x) \leq x + x^2$ for all $x > 0$.
Finally, for \( n \geq 10 \) \((N \geq 5)\) it holds that
\[
\log_\phi \left( \frac{\sqrt{5}}{3} (n + 2) \right) = \log_\phi n + \log_\phi \left( \frac{\sqrt{5}}{3} \right) + \log_\phi \left( 1 + \frac{2}{n} \right)
\leq \log_\phi n - \frac{3}{5} + \frac{2}{n \log \phi} + \frac{4}{n^2 \log \phi} < \log_\phi n - \frac{1}{5},
\]
and
\[
\log_\phi \left( \frac{\sqrt{5}}{3} (n + 2) \right) > \log_\phi n + \log_\phi \left( \frac{\sqrt{5}}{3} \right) > \log_\phi n - \frac{4}{5}.
\]
By putting all these bounds together we obtain that \( N < \log_\phi n < N + 1 \), which completes the proof.

**Theorem 10** If \( T_N^F \) is the \( N \)-th Fibonacci tree, then \( \text{imb}(T_N^F) = F_N \), where \( F_N \) is \( N \)-th Fibonacci number.

**Proof** Clearly \( \text{imb}(T_N^F) = F_N \) for \( N = 1, 2 \).

Let \( N \geq 3 \) and proceed by induction. By construction, we know that the grandchildren of the root of \( T_N^F \) are exactly the roots of \( T_{N-1}^F \) and \( T_{N-2}^F \), and they all have the same color in any 2-coloring of \( T_N^F \). Therefore,
\[
\text{imb}(T_N^F) = \text{imb}(T_{N-1}^F) + \text{imb}(T_{N-2}^F) = F_{N-1} + F_{N-2} = F_N.
\]
Throughout the remainder of the paper we denote by \( \text{parent}(v) \) a vertex directly above \( v \) in \( T_N^F \). For consistency, we assume that if \( r \) is a root of \( T_N^F \), then \( \text{parent}(r) \) is properly defined so that the predicate \( \text{parent}(r) \in A \) is false for any set \( A \).

Observe that the set of vertices which are the roots of all Fibonacci subtrees is defined exactly by \( C_1(T_N^F) \), since the root of \( T_N^F \) is in \( C_1(T_N^F) \). Therefore, using Proposition 1 we can modify any red-blue-yellow \((k, l)\)-decomposition in such a way that \( Y \cap C_1(T_N^F) = \emptyset \):

**Lemma 5** Let \((R, B, Y)\) be a red-blue-yellow \((k, l)\)-decomposition of a Fibonacci tree \( T_N^F \). There exists \( 0 \leq k' \leq k + 2l \) such that we can construct a red-blue-yellow \((k', l)\)-decomposition \((R', B', Y')\) of \( T_N^F \) with \( Y' \cap C_1(T_N^F) = \emptyset \).

**Proof** First, let us define \( Y' = \{ v \in C_2(T_N^F) : v \in Y \text{ or parent}(v) \in Y \} \), i.e. let us “push down” the yellow color from roots of Fibonacci subtrees in \( T_N^F \) to their children. Clearly, \( Y' \cap C_1(T_N^F) = \emptyset \) and moreover \( |Y'| \leq |Y| \) since the construction implicitly defines a function from \( Y \) into \( Y' \).

Next, we define \( R' \) and \( B' \) to be as following:

- if \( v \in C_2(T_N^F) \), \( v \notin Y \) and \( \text{parent}(v) \notin Y \), then we preserve the color: if \( v \in R \), then we put \( v \) in \( R' \), otherwise we put it in \( B' \);
- if \( v \in C_1(T_N^F) \) and \( v \notin Y \), then we also preserve the color of \( v \);
- if \( v \in C_1(T_N^F) \), \( v \in Y \) and \( \text{parent}(v) \notin Y \), then we use the color different than in the parent: if \( \text{parent}(v) \in R \), then we add \( v \) to \( R' \), otherwise we add \( v \) to \( B' \);
- finally, if \( v \in C_1(T_N^F) \), \( v \in Y \) (so by definition the only child of \( v \) is in \( Y' \)) and \( \text{parent}(v) \in Y \), we put \( v \) in \( R' \).

Intuitively, this whole procedure “extends” sets of blue and red vertices to the previously yellow vertices \( v \in Y \cap C_1(T_N^F) \): if \( \text{parent}(v) \) was blue (respectively, red), we use red (respectively, red) to color \( v \). And if we are certain that both a parent and an only child of a vertex in \( C_1(T_N^F) \) will be in \( Y' \) (the last case above), then we just assign it to \( R' \), as it will be surrounded only by the vertices from \( Y' \) (in fact, it could be also added to \( B' \) instead).

Note that \( |R'| \leq |R| + |Y| \) as only in the last two cases we can add some vertices to \( R' \) which were not in \( R \) – but this means that these vertices were in \( Y \). Additionally, \( |B'| \geq |B| - |Y'| \geq |B| - |Y| \), as the only vertices which are in \( B \setminus B' \) are exactly the ones which are added to \( Y' \).
Finally, we verify that indeed \( (R', B', Y') \) or \( (B', R', Y') \) is a red-blue-yellow \((k', l)\)-decomposition of \( T_N^F \) for some
\[
k' = ||R'|-|B'|| \leq ||R|-|B|+2||Y'|| \leq k+2l,
\]
because \( R' \) and \( B' \) are independent sets and \( |Y'| \leq |Y| \leq l \).

To see that \( R' \) is an independent set, let us note that if there were some \( v, \text{parent}(v) \in R' \), then \( v, \text{parent}(v) \notin B \), since we never swap between red and blue colors in the procedure above. Thus we could have only five cases:

- \( v, \text{parent}(v) \in Y - \) but since one of them is in \( C_2(T_N^F) \), it has to be also in \( Y' \), not in \( R' \),
- \( v \in Y \cap C_1(T_N^F), \text{parent}(v) \in R - \) then \( v \) would be rather in \( B' \), not in \( R' \).
- \( v \in Y \cap C_3(T_N^F), \text{parent}(v) \in R - \) thus \( v \) would be in \( Y' \), not in \( R' \).
- \( v \in R \cap C_1(T_N^F), \text{parent}(v) \in Y - \) then by construction \( \text{parent}(v) \) would be in \( Y' \), not in \( R' \).
- \( v \in R \cap C_2(T_N^F), \text{parent}(v) \in Y - \) and \( v \) would be in \( Y' \), not in \( R' \).

Either way, we would obtain a contradiction. Note that the same argument can be also used to show that \( B' \) is an independent set, which concludes the proof.

Now, we turn to the computation of the imbalances of red-blue parts of Fibonacci trees. In order to do that, we have to recall some theory of Fibonacci numbers. In particular, by the Zeckendorf theorem \[23\] we know that any number \( n \in \mathbb{N} \) can be written as
\[
n = \sum_{j=1}^\infty z_j F_j \text{ for } z_j \in \{0, 1\} \text{ such that}
\]
(A) no two consecutive \( z_j \) are equal to 1, i.e. \( z_j z_{j+1} = 0 \) for all \( j \).
(B) this representation is unique and it is the decomposition with the smallest value of \( \sum_j |z_j| \).
(C) it can be constructed by greedy subtraction of the largest possible Fibonacci numbers from \( n \) until we reach 0.

We call a vector \( (z_j)_{j=1}^\infty \) such that \( (A)-(C) \) holds the Zeckendorf representation of \( n \) and denote it by \( Z(n) \).

In fact, we can generalize the property \( (B) \) to \( (B') \) and show that for any \( n = \sum_j z'_j F_j \) with \( z'_j \geq 0 \) it has to be the case that
\[
\sum_j |z'_j| \geq \sum_j |z_j| \text{ for } Z(n) = (z'_j)_{j=1}^\infty.
\]

Without loss of generality we trim \( Z(n) \) to a finite number of positions with a leading one since \( F_i > n \) implies trivially \( z_i = 0 \). For example, \( Z(F_N) = (0, \ldots, 0, 1) \), i.e. it has a single one on \( N \)-th position and \( Z \left( \frac{F_N}{2} \right) = (\ldots, 1, 0, 0, 1) \), i.e. it has ones only on every third position ending at \( N - 2 \) as it is true that \( \frac{F_N}{2} = F_{N-2} + \frac{F_{N-1}}{2} \).

Now we may proceed to the lemma which ties tree decompositions and Fibonacci numbers:

**Lemma 6** For any red-blue-yellow \((k, l)\)-decomposition \((R, B, Y)\) of \( T_N^F \) such that \( Y \cap C_1(T_N^F) = \emptyset \) and \( k \geq 0 \) it holds that every connected component \( T_i \) of \( T_N^F \) \([R \cup B]\) has
\[
imbh(T_i) = |Y_i| + \sum_{j=1}^r z_{i,j} F_j
\]
where \( r \in [1, N], Y_i = \{ v \in Y : \text{parent}(v) \in V(T_i) \}, z_{i,j} \in \{-1, 0, 1\} \text{ for all } i \text{ and } j = 1, 2, \ldots , N \) and \( \sum_{j=1}^N |z_{i,j}| \leq |Y_i| \).

**Proof** The intuition behind the theorem is straightforward: since \( Y \cap C_1(T_N^F) = \emptyset \), it follows that each component \( T_i \) of \( T_N^F \) \([R \cup B]\) is a tree rooted in a vertex \( u_i \) that is also a root of some Fibonacci tree. It may be a Fibonacci tree itself – but even if it is not, then it means it has some Fibonacci subtrees removed, which are cut off exactly in vertices from \( Y_i \). But then every \( v \in Y_i \) has children that are also roots of some Fibonacci subtrees.

Now, let us define \( e \) as a 2-coloring of \( T_N^F \) and \( c_i \) as a 2-coloring of \( T_i \), such that all their respective roots have color 1. From now on, \( C_1(T_N^F) \) and \( C_2(T_N^F) \) are defined with respect to \( e \).
Finally, we can replace the last element using the fact that the definition of the Zeckendorf representation

\[ \text{imb}(T_i) = |Y_1| + F_n - F_{n-1} - F_{n-2} = 2. \]

Figure 3: An example Fibonacci tree \( T_n^F \) with \( |Y_1| = 2 \) and the imbalance of the top tree.

Observe that it has to be the case that \( e(u_i) = 1 \), since every root of a Fibonacci subtree is at the even distance from the root of the Fibonacci tree \( T_N^F \). Thus, for all \( v \in V(T_i) \) it holds that \( c_i(v) = c(v) \).

Moreover, every leaf \( v \) of \( T_i \) also has \( c_i(v) = 1 \), as either it is a leaf of \( T_N^F \) (thus \( c(v) = 1 \)), or it is a parent of some vertex from \( Y_i \). However, in the latter case \( Y \cap C(Y_i) = \emptyset \) implies also that \( c(v) = 1 \), as the vertices from \( Y_i \) always get color 2.

Finally, we note that to get \( \text{imb}(T_i) \) we have to:

- add the imbalance of a Fibonacci subtree rooted in \( u_i \), i.e. \( T_r^F \) for some \( r \in [1, N] \),
- remove the imbalance of some Fibonacci subtrees of \( T_r^F \) (i.e. the ones rooted in children of all \( v \in Y_i \) below \( u_i \)) – note that for each vertex from \( Y_i \) we remove its two subtrees \( T_{j-1}^F \) and \( T_{j-2}^F \) with total imbalance equal to \( \text{imb}(T_{j-1}^F) + \text{imb}(T_{j-2}^F) = F_j \),
- add the size of \( |Y_i| \) – vertices from \( Y_i \) were taken into account when adding the imbalance of the whole tree in the first case above, however, they are not in \( T_i \). Note that their removal will increase the imbalance since they are always in \( C(Y_i) \).

See also an example in Figure 3.

Overall, if the imbalance of a Fibonacci subtree of \( T_N^F \) rooted in \( u_i \) is equal to \( F_r \) for some \( r \), then we get that \( \text{imb}(T_i) = F_r - \sum_{j=1}^{r-1} z'_{i,j} F_j + |Y_i| \) for some \( z'_{i,j} \in \mathbb{N} \) with \( \sum_{j=1}^{r-1} |z'_{i,j}| \leq |Y_i| \). Here each \( z'_{i,j} \) denotes the number of Fibonacci subtrees \( T_j^F \) which had to be cut out from \( T_r^F \) to obtain exactly \( T_i \).

Finally, we can replace the last element using the fact that the definition of the Zeckendorf representation \( Z(\sum_{j=1}^{r-1} z_{i,j} F_j) = (-z_{i,r} Y_{r-1}^{-1}) \) implies \( \sum_{j=1}^{r-1} z_{i,j} F_j = \sum_{j=1}^{r-1} -z_{i,j} F_j \) for some \( z_{i,j} \in \{-1, 0\} \). And from the property \( (B') \) it follows that \( \sum_{j=1}^{r-1} |z_{i,j}| \leq \sum_{j=1}^{r-1} |z'_{i,j}| \leq |Y_i| \). Therefore, it is sufficient to set \( z_{i,r} = 1 \) to conclude the proof.

Lemma 7 If there exists a red-blue-yellow \((k, l)\)-decomposition \((R, B, Y)\) of \( T_N^F \) such that \( |Y| = l \), \( Y \cap C(Y_i) = \emptyset \) then there exists a value \( y \in [0, l] \) and a vector \( z = (z_j)_{j=1}^N \) with \( z_j \in \{0, 1\} \), \( \sum_{j=1}^N |z_j| \leq 2l + 1 \), and

\[
\frac{F_N - k + l}{2} = y + \sum_{j=1}^N z_j F_j.
\]

Proof The required decomposition exists if and only if there exists a set \( Y \) with \( l \) vertices such that we can find for connected components \( T_i \) of \( T_N^F \) certain 2-coloring \( c \) which induces a partition into \( R \) and \( B \) such that \( |R| - |B| = k \).
From Lemma 6 we know that for every $T_i$ we can write $imb(T_i) = |Y_i| + \sum_{j=1}^{N} z_{i,j} F_j$ such that $z_{i,j} \in \{-1, 0, 1\}$ and $\sum_{j=1}^{N} |z_{i,j}| \leq |Y_i| + 1$. Thus, 

$$k = |R| - |B| = \sum_{i=1}^{l+1} a_i \cdot imb(T_i) = y + \sum_{i=1}^{l+1} \sum_{j=1}^{N} a_i z_{i,j} F_j.$$ 

for some $y = \sum_{i=1}^{l+1} a_i |Y_i| \in [-l, l]$ and some $a_i \in \{-1, 1\}$.

Intuitively, we add or subtract imbalances of $T_i$ depending on the fact whether the root of $T_i$ has color 1 or 2 in $c$. In particular, $a_i = -1$ if and only if the root of $T_i$ has color 2 in $c$. Note that from $|Y| = l$ we know that $T_N^F[R \cup B] = T_N^F \setminus Y$ has exactly $l + 1$ connected components (some possibly empty).

Now let us consider a coloring $c'$ of $T_N^F$ where the root has color 1. Note that all subtrees $T_i$ are rooted in vertices with color 1 in $c'$, and all vertices from $Y$ have color 2 in $c'$. Therefore, 

$$F_N + l = imb(T_N^F) + |Y| = \sum_{i=1}^{l+1} imb(T_i) = |Y| + \sum_{i=1}^{l+1} \sum_{j=1}^{N} z_{i,j} F_j,$$

where the last equality follows from Lemma 6.

Now, by subtracting the last two equations we obtain 

$$\frac{F_N - k + l}{2} = \frac{|Y| - y}{2} + \sum_{i=1}^{l+1} \sum_{j=1}^{N} a_i' z_{i,j} F_j,$$

where we used the fact that $a_i' = \frac{1-a_i}{2} \in \{0, 1\}$ for all $i = 1, 2, \ldots, l + 1$.

Now observe that $\sum_{j=1}^{N} F_j \sum_{i=1}^{l+1} a_i' z_{i,j}$ is a sum of Fibonacci numbers, i.e. each $F_j$ appears exactly $\sum_{i=1}^{l+1} a_i' z_{i,j}$ times. The total count of Fibonacci numbers in this sum does not exceed 

$$\sum_{i=1}^{l+1} \sum_{j=1}^{N} |a_i' z_{i,j}| \leq \sum_{i=1}^{l+1} \sum_{j=1}^{N} |z_{i,j}| \leq \sum_{i=1}^{l+1} (|Y_i| + 1) \leq |Y| + l + 1 = 2l + 1,$$

since we know from Lemma 6 that $\sum_{j=1}^{N} |z_{i,j}| \leq |Y_i| + 1$ for every $i$.

Therefore, by Zeckendorf theorem there exists a vector $z' = (z'_j)_{j=1}^{N}$ with $z'_j \in \{0, 1\}$ so that $\sum_{j=1}^{N} |z'_j| \leq 2l + 1$ and 

$$Z \left( \sum_{i=1}^{l+1} \sum_{j=1}^{N} a_i' z_{i,j} F_j \right) = z'.$$

This completes the proof.

**Lemma 8** For $N \geq 96$, any $k \in [0, F_{N/2-1}]$, $l \in \left[1, \frac{N}{48} - 1\right]$ there does not exist a value $y \in [0, l]$ and a vector $z = (z_j)_{j=1}^{N}$ with $z_j \in \{-1, 0, 1\}$ and $\sum_{j=1}^{N} |z_j| \leq 2l + 1$ such that 

$$\frac{F_N - k + l}{2} = y + \sum_{j=1}^{N} z_j F_j.$$
Proof It is well known that for any \( N \geq 4 \),
\[
\frac{F_N}{2} = \sum_{i=0}^{\lfloor \frac{N-2}{2} \rfloor} F_{N-2i}.
\]

From now on let us denote \( K = F_{N/2-1} \) and \( L = \frac{N}{48} - 1 \). In fact, we would like to prove a slightly stronger claim: that any number \( S \in \left[ \frac{F_N-K-1}{2}, \frac{F_N+L}{2} \right] \) (a range encompassing all possible values of \( \frac{F_N-K+1}{2} - y \) since \( y \leq 1 \)) cannot be represented using a sum or difference of at most \( 2L+1 \) Fibonacci numbers.

Let \( S' = S - \frac{F_N}{2} \). By assumption it holds that \( L < K = F_{N/2-1} \), so \( |S'| \leq \frac{K+L}{2} < F_{N/2-1} \) and therefore \( Z(|S'|) \) cannot have ones on positions \( \left[ \frac{N}{2} - 1, N \right] \). Let us take positions \( \left[ 1, \frac{N}{2} - 1 \right] \) of \( Z \left( \frac{F_N}{2} \right) \), equivalent to some number \( S'' \) and observe that \( S'' + S' \geq 0 \) – and, additionally, it also cannot have ones on positions \( \left[ \frac{N}{2} - 1, N \right] \). Therefore, we are certain that \( Z(S) \) has exactly the same values as \( Z \left( \frac{F_N}{2} \right) \) at least on positions \( \frac{N}{2} + 1, \ldots, N \).

Note that we can split \( Z \left( \frac{F_N}{2} \right) \) into a sequence of \( \left[ \frac{N-1}{2} \right] \) blocks of length 3 equal to \((0,1,0)\). For convenience, but with a slight abuse of notation from now on we allow for a constant number of leading zeros in the Zeckendorf representations. Combining the last two facts we get that \( Z(S) = Z \left( \frac{F_N}{2} + S' \right) \) contains a sequence of \( \frac{N}{6} \) consecutive blocks \((0,1,0)\) between positions \( \frac{N}{2} + 1 \) and \( N \).

Now let us define \( S_1 = \sum_{j=1}^{N} F_j \cdot 1_{j \geq 0} \) and \( S_2 = \sum_{j=1}^{N} F_j \cdot 1_{j < 0} \). Clearly, \( S = S_1 - S_2 \) and \( Z(S_1) \) and \( Z(S_2) \) have both in total at most \( 2L+1 \) ones. Therefore, to prove this lemma it is sufficient to show that if \( Z(S) \) consists of \( \frac{N}{6} \) consecutive blocks \((0,1,0)\) and \( Z(S_2) \) has only \( 2L+1 \) ones, then \( Z(S_1) \) cannot have at most \( 2L+1 \) ones. From now on we will concentrate only on these sections of the vectors \( Z(S) \) and \( Z(S_2) \), and the possible carry operations in the addition \( Z(S) + Z(S_2) = Z(S_1) \).

Let us write both \( Z(S) \) and \( Z(S_2) \) as blocks of length 3 and denote \( \ell \)-th block of \( Z(S_2) \) (and also, by association, the respective block of \( Z(S) \)) as type A if it is non-zero, and as type B otherwise. Note that there can be at most \( 2L+1 \) blocks of type A since \( Z(S_2) \) has at most \( 2L+1 \) ones.

Now consider how many blocks of type B can be influenced by a carry from blocks of type A. We have to analyze not only the forward carry but also the backward one, for example, in Zeckendorf representation we have \( Z(3) = (0,0,0) \) but \( Z(3+3) = (0,1,0;0,1,0) \).

The proof for the forward carry is simpler: suppose we consider the \( \ell \)-th block of type A, which starts with a position \( j \), and that the \((i+1)\)-th block is of type B. Then we know that the number \( S^*_2 \) indicated only by the blocks up to \( i-1 \) cannot exceed \( F_j - 1 \) by the definition of Zeckendorf representation. Similarly, for \( Z(S) \) the respective number \( S^* \) cannot exceed \( F_{j-1} - 1 \) since its \((i-1)\)-th block has to be equal to \((0,1,0)\). Now, the forward carry has three components:

- the forward carry from the sum of \( S^* \) and \( S^*_2 \) to the \( \ell \)-th block – which does not exceed \((F_j - 1) + (F_{j-1} - 1) < F_{j+1} \), so it can only be in the form \((1,0,0)\),
- the value of the \( i \)-th block of \( Z(S) \) – equal to \((0,1,0)\),
- the value of the \( \ell \)-th block of \( Z(S_2) \) – at most equal to \((1,0,1)\) since it does not have two consecutive ones.

In any way, the forward carry to the \((i+1)\)-th block cannot exceed \((1,1,0)\). However, since the \((i+1)\)-th blocks of \( Z(S) \) and \( Z(S_2) \) are \((0,1,0) \) and \((0,0,0)\), respectively (since it is of type B), there is no forward carry to the \((i+2)\)-th block and beyond.

That leaves us with proving that backward carry from a block of type A cannot influence too many blocks of type B. First, we note that \( Z(S_2) \) by the property (A) of the Zeckendorf representation does not have two consecutive ones. Thus, the only combinations available when we sum the rightmost blocks of type A (i.e. the ones which do not have blocks of type A to the right) are:

1. \((0,1,0) + (1,0,0) = (0,0,1)\),
2. \((0,1,0) + (0,1,0) = (0,0,1)\) with a backward carry \((0,0,1)\).
Now, observe that if the block to the left is also of type A, then a respective block from \( S \) to the rightmost block. And regardless of the value of the appropriate block of \( Z(S_2) \), the total sum of the blocks and the backward carry cannot generate any further backward carry.

Finally, note that the aforementioned forward carry resulting from backward carry appears in the block which has to be equal to \((0, 0, 1)\) (as it has to be the second case above), so it turns it into \((1, 0, 1)\) and it does not generate any future carries.

Therefore, the only possible backward carry from the block of type A to the block of type B has to be in the form \((0, 0, 1)\).

However, this will be combined with a block \((0, 1, 0)\) from \( Z(S) \) – thus, the sum of the blocks from \( Z(S), Z(S_2) \) and the backward carry again cannot result in any further backward carry.

Thus, we proved that any block of type A can affect up to one block of type B to the left and one block of type B to the right.

If there are only \( 2l + 1 \) blocks of type A, then at most \( 3(2l + 1) \) blocks of type A and B can be modified – and therefore at least \( \frac{N}{6} - 3(2l + 1) \) blocks of the form \((0, 1, 0)\) of \( Z(S - S_2) \) are identical to the respective blocks of \( Z(S) \) despite the addition of \( Z(S_2) \).

This means that \( Z(S - S_2) \) has to have at least \( \frac{N}{6} - 3(2L + 1) \) ones – which contradicts the assumption that \( Z(S - S_2) = Z(S_1) \) has at most \( 2L + 1 \) ones for \( L = \frac{N}{48} - 1 \).

**Theorem 11** For any \( N \geq 96 \), any \( k \in \left[0, F_{N/2-1} - \frac{N}{24} + 2\right] \) and any \( l \in \left[1, \frac{N}{48} - 1\right] \) there does not exist a red-blue-yellow \((k, l)\)-decomposition of \( T_N^F \).

**Proof** A direct implication of Lemma 7 and Lemma 8 is that there does not exist a red-blue-yellow \((k', l)\)-decomposition of \( T_N^F \) such that \( Y \cap C_1(T_N^F) = \emptyset \) for any \( k' \in [0, F_{N/2-1}] \) and \( l = \frac{N}{48} - 1 \).

Now, it follows from Lemma 5 that if there was a red-blue-yellow \((k, l)\)-decomposition of \( T_N^F \) for \( k \), then it would exist a red-blue-yellow \((k', l)\)-decomposition of \( T_N^F \) with \( Y \cap C_1(T_N^F) = \emptyset \) and \( k' \in [0, k + 2l] \subseteq [0, F_{N/2-1}] \) – thus, a contradiction.

Finally, we can proceed to the main theorem of this section, which shows the existence of the trees with constant degrees such that \( B_{BC}(K_n, T) = \max\{n, 2\lambda\} + \Omega(\log n) \):

**Theorem 12** There exists an infinite family of trees \( T_N^F \) on \( n = 3F_N - 2 \) vertices with \( \Delta(T_N^F) = 3 \) for \( N \geq 96 \) such that \( B_{BC}(K_n, T_N^F) \geq \max\{n, 2\lambda\} + \frac{1}{48} \log_2 n - 3 \) for \( \lambda = \left\lceil \frac{N}{24} \right\rceil \).

**Proof** Assume that \( l = \frac{N}{48} - 1 \). Then it always holds that \( 2\lambda - n + l \leq \lambda - 1 \) – so in Theorem 8 we need that there cannot be any red-blue-yellow \((k, l)\)-decomposition for \( k \in [0, 2\lambda - n + l] \). If \( N \geq 96 \), then it is easy to check that \( F_{N/2-1} = \frac{N}{24} + 2 > 2\lambda - n + l \).

Therefore, by combining Theorems 8 and 11 we get that \( B_{BC}(K_n, T_N^F) > 2\lambda + \frac{N}{48} - 1 \). To finish the proof it is sufficient to combine this result with two simple observations:

- by assumption it holds that \( 2\lambda \geq \max\{n, 2\lambda\} - 1 \),
- by Theorem 7 it is true that \( N = \lceil \log_2 n \rceil > \log_2 n - 1 \).

In fact, the theorem above can be proven in a more complicated, but also more general form, i.e. that \( B_{BC}(K_n, T_N^F) > 2\lambda + \frac{1}{48} \log_2 n - 2 \) for all \( \lambda \in \left[ \frac{n}{2} - \frac{1}{96} \log_2 n, \frac{n}{2} \right] \) – which gives us also an additive factor over \( \max\{n, 2\lambda\} \).

## 4 Conclusion

In this paper, we presented a new bound on \( B_{BC}(K_n, F) = \max\{n, 2\lambda\} + \Delta^2(F) \lceil \log n \rceil \) for forests \( F \) on \( n \) vertices. This is the first known upper bound depending on the maximum degree of \( F \), better than the existing ones when this parameter is small,
e.g. constant. We also showed that the same reasoning allows us to construct a linear algorithm for finding an $\lambda$-backbone coloring of $K_n$ with backbone $F$ within an additive error of $\Delta^2(F)\lceil \log n \rceil$ from the optimum. Finally, we proved that the bound is asymptotically tight, i.e. there exists a family of trees with $\Delta(T) = 3$ such that $BBC_{\lambda}(K_n, F) = \max\{n, 2\lambda\} + \Theta(\log(n))$.

However, there remain a couple of open problems that stem from our research. First, we can ask if the algorithm from Section 2 can be improved, i.e. does there exist an algorithm running in polynomial time which gives us a coloring using $BBC_{\lambda}(K_n, F) + o(\log(n))$ colors. Or maybe it is the case that finding a $\lambda$-backbone coloring such that $BBC_{\lambda}(K_n, F) + f(n, \Delta(F))$ for some function $f$ is $NP$-hard – either for $\Delta(F)$ constant or provided as a part of input.

Second, there arises a natural question of how to extend our results to other classes of graphs. An obvious extension would be an analysis for a class of split graphs, i.e. graphs whose vertices can be partitioned into a maximum clique $C$ (of size $\omega(G) = \chi(G)$) and an independent set $I$. A simple application of Theorem 7 gives us that $BBC_{\lambda}(G, F) \leq \lambda + \max\{\chi(G), 2\lambda\} + \Delta^2(F)\lceil \log \chi(G) \rceil$, as we can first solve the problem restricted to $C$, find a $\lambda$-backbone coloring using colors from the set $[1, \max\{\chi(G), 2\lambda\} + \Delta^2(F)\lceil \log \chi(G) \rceil]$ and then color all vertices in $I$ with a color $\lambda + \max\{\chi(G), 2\lambda\} + \Delta^2(F)\lceil \log \chi(G) \rceil$.

However, it was proved before that

**Theorem 13 (Salman, [24])** If $G$ is a split graph and $T$ is its spanning tree, then

$$ BBC_{\lambda}(G, T) \leq \begin{cases} 
1 & \text{for } \chi(G) = 1, \\
\lambda + 1 & \text{for } \chi(G) = 2, \\
\lambda + \chi(G) & \text{for } \chi(G) \geq 3.
\end{cases} $$

This bound is tight.

Although our bound is more general, as it applies to all backbones $H$ as long as $H[C]$ is a forest, a simple comparison of both bounds shows that for trees (and forests) our bound is worse in all cases, even for small $\lambda$ and large $\chi(G)$. This is mainly because we assign a single color to $I$ naively instead of trying to optimize it. Still, we can pose a question: is there a way to amend our approach to make it suitable also for split graphs and improve the existing bound at least for some subset of graphs?

**REFERENCES**

[1] Broersma H, Fomin F, Golovach P, Woeginger G. Backbone colorings for graphs: tree and path backbones. Journal of Graph Theory 2007;55(2):137–152.

[2] Bondy JA, Murty USR. Graph Theory, vol. 244 of Graduate Texts in Mathematics. Springer; 2008.

[3] Havet F, King A, Liedloff M, Todinca I. (Circular) backbone colouring: Forest backbones in planar graphs. Discrete Applied Mathematics 2014;169:119–134.

[4] Janczewski R, Turowski K. The backbone coloring problem for bipartite backbones. Graphs and Combinatorics 2015;31(5):1487–1496.

[5] Broersma H, Marchal B, Paulusma D, Salman ANM. Backbone colorings along stars and matchings in split graphs: their span is close to the chromatic number. Discussiones Mathematicae Graph Theory 2009;29(1):143–162.

[6] Janczewski R, Turowski K. The computational complexity of the backbone coloring problem for bounded-degree graphs with connected backbones. Information Processing Letters 2015;115(2):232–236.
[7] Miškuf J, Škrekovski R, Tancer M. Backbone colorings and generalized Mycielski graphs. SIAM Journal on Discrete Mathematics 2009;23(2):1063–1070.

[8] Araujo C, Araujo J, Silva A, Cezar A. Backbone Coloring of Graphs with Galaxy Backbones. Electronic Notes in Theoretical Computer Science 2019;346:53–64.

[9] Miškuf J, Škrekovski R, Tancer M. Backbone colorings of graphs with bounded degree. Discrete Applied Mathematics 2010;158(5):534–542.

[10] Calamoneri T. The L(h,k)-labelling problem: an updated survey and annotated bibliography. The Computer Journal 2011;54(8):1344–1371.

[11] Broersma H. A general framework for coloring problems: old results, new results, and open problems. In: Akiyama J, Baskoro ET, Kano M, editors. Indonesia-Japan Joint Conference on Combinatorial Geometry and Graph Theory, vol. 3330 of Lecture Notes in Computer Science Springer; 2003. p. 65–79.

[12] Čada R, Ekstein J, Holub P, Togni O. Radio labelings of distance graphs. Discrete Applied Mathematics 2013;161(18):2876–2884.

[13] Bantva D, Vaidya S, Zhou S. Radio number of trees. Discrete Applied Mathematics 2017;217:110–122.

[14] Korže D, Shao Z, Vesel A. New results on radio k-labelings of distance graphs. Discrete Applied Mathematics 2022;319:472–479.

[15] Fotakis D, Spirakis P. A Hamiltonian approach to the assignment of non-reusable frequencies. In: Arvind V, Ramanujam R, editors. 18th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 1998), vol. 1530 of Lecture Notes in Computer Science Springer; 1998. p. 18–29.

[16] Damaschke P, Deogun J, Kratsch D, Steiner G. Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm. Order 1991;8:383–391.

[17] Chang G, Kuo D. The L(2,1)-labeling problem on graphs. SIAM Journal on Discrete Mathematics 1996;9(2):309–316.

[18] Papadimitriou C, Yannakakis M. The traveling salesman problem with distances one and two. Mathematics of Operations Research 1993;18(1):1–11.

[19] Karpinski M, Lampis M, Schmied R. New inapproximability bounds for TSP. Journal of Computer and System Sciences 2015;81(8):1665–1677.

[20] Adamaszek A, Mnich M, Paluch K. New approximation algorithms for (1, 2)-TSP. In: Chatzigiannakis I, Kaklamanis C, Marx D, Sannella D, editors. 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), vol. 107 Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik; 2018. p. 9:1–9:14.

[21] Turowski K. Optimal backbone coloring of split graphs with matching backbones. Discussiones Mathematicae Graph Theory 2015;35(1):157–169.

[22] Araujo J, Cezar AA, Silva A. On the existence of tree backbones that realize the chromatic number on a backbone coloring. Journal of Graph Theory 2017;85(4):808–813.

[23] Knuth D. Fibonacci multiplication. Applied Mathematics Letters 1988;1(1):57–60.

[24] Salman ANM. \( \lambda \)-backbone coloring numbers of split graphs with tree backbones. In: Proceeding of The Second IMT-GT 2006 Regional Conference on Mathematics, Statistics and Applications School of Mathematical Sciences, Universiti Sains Malaysia; 2006. p. 43–47.