# Approximations for $\lambda$-coloring in treewidth $k$ graphs, outerplanar graphs and split graphs

Simone Bianco

*Sapienza Università di Roma, Italy*

December 7, 2024

**Abstract**

This essay is based on the article written by Bodlaender, Kloks, Tan, et al. [BKT+04]. The contents of in the article are discussed in a more in-depth way, showing graphical examples and providing access to a non-expert reader.

## Contents

# 1 Introduction

In the context of radio frequency assignment, the goal is to allocate radio frequencies to transmitters at various locations while avoiding interference. This problem is analogous to graph coloring, where the vertices of the graph represent the transmitters, and the edges indicate potential interferences between adjacent vertices. The radio frequencies are represented as non-negative integers, often referred to as "colors".

To model this problem, Griggs and Yeh [GY92] introduced the **L(2, 1)-labeling problem**, where the transmitters are modeled as nodes of a graph whose edges depend on the capability of communication between the transmitters. The problem involves assigning colors to nodes under specific constraints:

1. Nodes that are very close to each other (at a distance of 1) must have a color value differing by at least two.
2. Nodes that are relatively close (at a distance of 2) must have a color value differing by at least one.

The problem is thus equivalent to defining an assignment, known as **$\lambda$-coloring**, from to each node in the network to integers from the set $\{0, \ldots, \lambda\}$, known as the *span*, ensuring the two constraints are satisfied. The objective of $\lambda$-coloring is to find the optimal (or near-optimal) bounds of the minimal value $\lambda_{2,1}$ that gives a span capable of coloring a graph. Over the years, the problem has been generalized to the $L(p, q)$-coloring, where $a$ and $b$ are two arbitrary non-negative integers.

In this essay, we'll focus on the most studied values for this problem: $\lambda_{2,1}, \lambda_{1,1}$ and $\lambda_{0,1}$. In particular, we'll focus on studying bounds for these colorings on **graph classes**, i.e. sets of graphs with a given property.

**Definition 1.** *Let $G$ be a graph and $p, q$ be two non-negative integers. A $\lambda$-coloring on $G$ is a function $f : V(G) \to \{0, \ldots, \lambda\}$. The $\lambda$-coloring satisfies the $L(p, q)$-constraints if*

- *For each $u, v \in V(G)$ such that $\mathrm{dist}(u, v) = 1$ it holds that $|f(u) - f(v)| \geq p$*
- *For each $u, v \in V(G)$ such that $\mathrm{dist}(u, v) = 2$ it holds that $|f(u) - f(v)| \geq q$*

*The minimum value $\lambda$ for which $G$ admits a $\lambda$-coloring satisfying the $L(p, q)$-constraint is denoted by $\lambda_{p,q}(G)$.*

The usual parameter used in these bounds is $\Delta$, i.e. the maximum degree of any node in the graph. For any general graph, the easiest

lower bounds can be obtained from the tree graph $K_{1,\Delta}$, consisting of a tree with one root that has $\Delta$ children. Since any graph with maximum degree $\Delta$ contains $K_{1,\Delta}$ as a subgraph, the following lower bounds valid for any graph [GY92; BB92]:

$$\lambda_{2,1} \geq \Delta + 1 \qquad \lambda_{1,1} \geq \Delta \qquad \lambda_{0,1} \geq \Delta - 1$$

[GY92] also proved the general upper bound $\lambda_{2,1} \leq \Delta^2 + 2\Delta$, which was later improved to $\lambda_{2,1} \leq \Delta^2 + \Delta$ in [HRS08].

For some special classes of graphs, tight bounds for $\lambda(\mathcal{G})$ are known and can be computed efficiently. Here, new upper bounds are provided for graphs of treewidth $k$, outerplanar graphs and split graphs.

# 2 Graphs of treewidth $k$

In graph theory, the *treewidth* of an undirected graph is an integer number which specifies, informally, "how far" the graph is from being a "tree". The treewidth is defined in terms of *tree decomposition*.

**Definition 2.** *A tree decomposition of a graph $G$ is a tree $T$ with nodes $V(T) = \{X_1, \ldots, X_t\}$ where $X_i \subseteq V(G)$ for any $i$ that satisfies the following properties:*

1. *The set $X_1, \ldots, X_t$ is a cover of $V$, meaning that each graph node is contained in at least one tree node.*
2. *If $v \in X_i \cap X_j$ then all the tree nodes $X_h$ in the path from $X_i$ to $X_j$ contain $v$ as well.*
3. *For all graph edge $(u, v)$, there is a subset $X_i$ that contains $v, w$*

The *width* of a tree decomposition is the size of its largest set $X_i$ minus one. The *treewidth* of a graph $G$, written as $\mathrm{tw}(G)$, is the minimum width among all possible tree decompositions of $G$. In this context, the minus one factor in the width measure is considered in order to make any standard tree have treewidth 1.

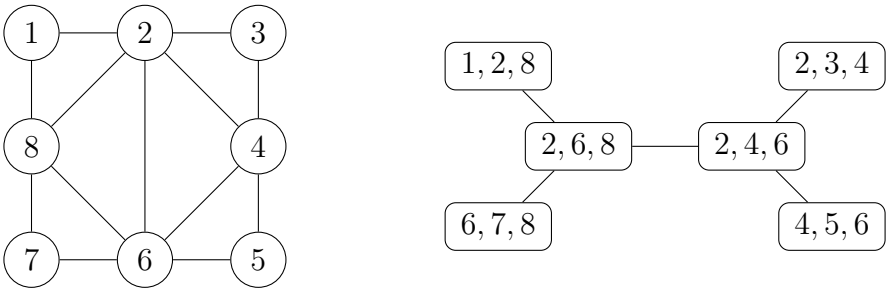



Figure 1: A graph of treewidth 2 and his minimum width tree decomposition.

Through this definition, finding the treewidth of a graph is no easy task. An easier way to find the treewidth of a graph is through *k-trees*, a generalization of the concept of standard trees. This class of graphs is inductively defined as follows.

**Definition 3.** *Given an integer $k > 0$, a graph $G$ with $n$ nodes is said to be a $k$-tree if one of the two conditions holds:*

- *$G$ is the complete graph $K_{k+1}$*
- *$G$ can be built from a $k$-tree $G'$ with $n-1$ nodes by adding a new node and making it adjacent to exactly all the vertices of a $k$-clique in $G'$*

*Any graph $H$ that is a subgraph of a $k$-tree is called a partial $k$-tree.*

We notice that $k$-trees are exactly the maximal graphs with a treewidth of $k$ – hence their name – where "maximal" means that no more edges can be added without increasing their treewidth. In fact, it can be shown that a graph is a partial $k$-tree if and only if its treewidth is at most $k$ [Nes08]. To show this result, we first state the following trivial lemma.

**Theorem 1.** *A graph $G$ is a partial $k$-tree if and only if* $\mathrm{tw}(G) \leq k$

*Proof.* $\Rightarrow$) Let $G = (V, E)$ be a partial $k$-tree and let $H = (V, E')$ be the $k$-tree that contains $G$ as a subgraph. Clearly, we have that $\mathrm{tw}(G) \leq \mathrm{tw}(H)$. We will show that $\mathrm{tw}(H) \leq k$. We proceed by induction on the number of vertices of $H$.

If $H$ has $k+1$ vertices then $\mathrm{tw}(H) = k$ since a single node with all the vertices is a tree decomposition of $H$ of width $k$. Assume now that $H$ has more than $k+1$ vertices. By definition of $k$-tree, there is a node $v \in V$ such that $H[V - \{v\}$ is a $k$-tree and $v \cup N_H(v)$ induce a $(k+1)$-clique $K$ in $H$. By inductive hypothesis, the induced subgraph $H[V - \{v\}]$ has a tree decomposition $T = (\{X_1, \ldots, X_t\}, F)$ of width at most $k$.

Since $K - \{v\}$ is a $k$ clique in $H[V - \{v\}]$, by the previous lemma there must be a set $X_i$ such that $K - \{v\} \subseteq X_i$. Let $X_{t+1} = K$ and let $T' = (\{X_1, \ldots, X_t, X_{t+1}\}, F \cup \{i, t+1\})$. Then, since $|K| - 1 = k$, $T'$ is a tree decomposition of $G$ with width at most $k$.

$\Leftarrow$) Again, we proceed by induction. Let $G = (V, E)$. If $G$ has $k+1$ vertices then the $k$-tree $K_{k+1}$ trivially contains $G$.

Assume now that now that $G$ has more than $k+1$ vertices. Let $T = (\{X_1, \ldots, X_t\}, F)$ be a tree decomposition of $G$ of width at most $k$. Let $X_\ell$ be a leaf node of $T$ and let $X_i$ be its only neighbor in $T$. We can assume that $X_\ell \not\subseteq X_i$ since otherwise by removing $X_\ell$ from $T$ we still get a tree decomposition of $G$ with width at most $k$. Moreover,

this assumption also allows us to assume that $X_\ell - X_i = \{v\}$: if not, $X_\ell$ can be divided into several tree nodes such that the resulting new leaf node satisfies this requirement.

Since $X_\ell$ is a leaf node and $X_\ell - X_i = \{v\}$, every adjacent vertex of $v$ must be inside $X_\ell$ and $X_i$. Moreover, since $|X_\ell| \leq k + 1$, $v$ has at most $k$ such neighbors. By inductive hypothesis, $G[V - \{v\}]$ is a partial $k$ tree. Let $H$ be the $k$-tree that contains $G[V - \{v\}]$. By construction of $H$, $X_i$ must be a $k$-clique, hence by adding $v$ to $H$ and by connecting it to all the vertices in $X_i$, we get a $k$-tree that contains $G$. □

**Corollary 1.** *A graph $G$ has treewidth $k$ if and only if $k$ is the smallest value for which $G$ is a partial $k$-tree.*

For instance, since in Section 2 we have a graph of treewidth 2, we know that it must be a partial 2-tree, meaning that we can construct a 2-tree that contains it. In fact, this graph is actually a full 2-tree and not only a partial one.

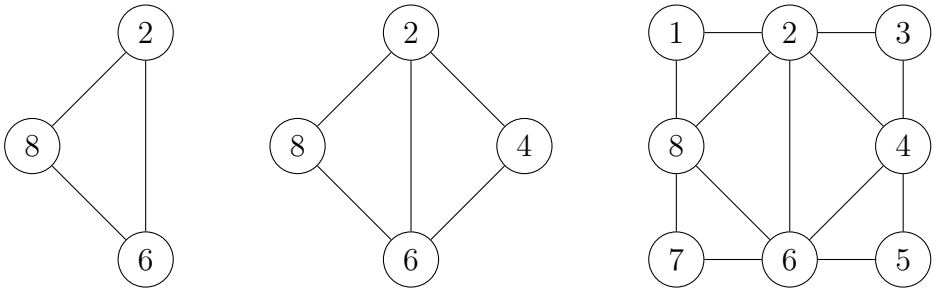



Figure 2: Steps 1, 2 and 6 of the inductive creation of the 2-tree in Section 2

We observe that $k$-trees are also exactly the *chordal graphs* whose maximal cliques all have size $k+1$ and whose minimal clique separators all have size $k$ [Pat86]. A clique separator is a set of vertices in a graph that, when removed, disconnects the graph into multiple components, while the separator itself forms a clique.

**Definition 4.** *A chordal graph is a graph in which all cycles of four or more vertices have a chord, i.e. an edge that is not part of the cycle but connects two vertices of the cycle.*

Equivalently, a chordal graph can be defined as a graph in which every induced cycle in the graph has exactly three vertices. For this reason, chordal graphs are also called *triangulated graphs* – again, see Section 2. Chordal graphs have a very special property: they always contain a perfect elimination sequence [Ros70].

**Lemma 1.** *A graph is chordal if and only if it contains a perfect elimination sequence.*

A *perfect elimination sequence* in a graph is an ordering of the vertices of the graph such that for all nodes $v$ it holds that $v$ and the neighbors of $v$ that occur after $v$ in the order form a clique.

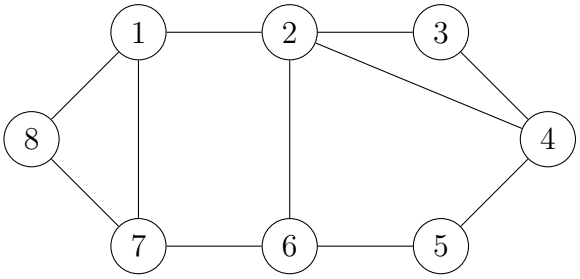
Since $k$-trees are a special type of chordal graph, any $k$-tree has a perfect elimination sequence. For instance, the ordering $(1, 3, 5, 7, 4, 2, 8, 6)$ is a perfect elimination sequence for the graph in Section 2.

Using the fact that any graph with treewidth $k$ is a partial $k$-tree and that any $k$-tree is a chordal graph, we can define the following algorithm able to $\lambda$-color that satisfies the $L(p, q)$-constraints for any graph of treewidth $k$.

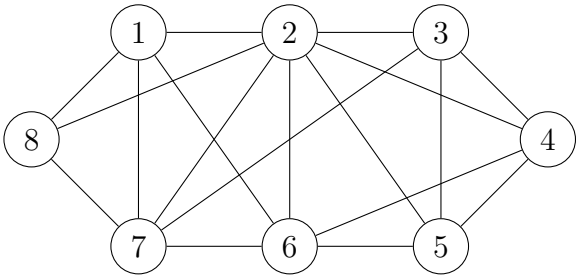
**function** ALGORITHM $1(G, \lambda, p, q)$
    Construct a $k$-tree $H$ that contains $G$ as a subgraph
    Construct a perfect elimination sequence $(v_1, \ldots, v_n)$ on $H$
    **for** $i = n, \ldots, 1$ **do**
        Color $v_i$ with the smallest available color in $\{0, \ldots, \lambda\}$
satisfying the $L(p, q)$-coloring constraints in $G$
    **end for**
**end function**

Figure 3: Algorithm for $\lambda$-Coloring Graphs of Treewidth $k$

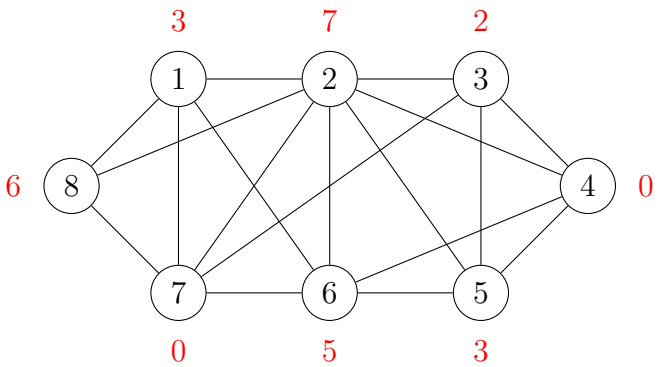
To understand how the algorithm works, consider the following graph with treewidth 3.




First, we construct the 3-tree that contains this graph as a subgraph.




A possible perfect elimination sequence for this 3-tree is given by $(8, 1, 7, 2, 6, 5, 3, 4)$. By coloring the sequence in reverse order while preserving the $L(2, 1)$-constraints we get the following color assignments.



We now prove that the algorithm always works with the following values.

**Theorem 2.** *Given any graph $G$ of treewidth $k$, the Algorithm 1 finds:*

1. *An $L(2,1)$-labeling using the set $\{0,\ldots,k\Delta+2k\}$.*
2. *An $L(1,1)$-labeling using the set $\{0,\ldots,k\Delta\}$.*
3. *An $L(0,1)$-labeling using the set $\{0,\ldots,k\Delta-k\}$.*

*Proof.* Since the Algorithm 1 clearly produces an $L(p,q)$-labeling, it suffices to show that the largest colors claimed by the theorem are enough. Given a vertex $v_i$, where $1 \leq i \leq n$, we count the number of colors that are forbidden for $v_i$ when it gets colored. In particular, due to the $L(p,q)$ constraints, can be one of three types of color forbidding nodes:

1. $\alpha$ already colored neighbors of $v_i$ in $G$.
2. $\beta$ already colored vertices that are at distance two from $v_i$ in $G$ and that have a neighbor in common with $v_i$ which has yet to be colored.
3. $\gamma$ already colored vertices that are at distance two from $v_i$ in $G$ and that have a neighbor in common with $v_i$ which has already been colored.

By construction of the perfect elimination sequence on a $k$-tree, each vertex has at most $k$ neighbors in $H$ that come after it in the sequence. We will show that each of the three cases account for one of such at most $k$ neighbors, i.e. that $\alpha+\beta+\gamma$.

Clearly, any type 1 vertex is one of such at most $k$ neighbors in $H$. Let $v_j$ be an already colored vertex at distance 2. If $v_j$ is a type 2 vertex, then we know that $h<i<j$ since the common neighbor $v_h$ has yet to be colored. Thus, by construction of the perfect elimination sequence, since $v_i$ and $v_j$ are adjacent to $v_h$ and they come after it in the sequence, they must be inside $v_h$'s neighbors that form a clique, concluding that $v_i$ and $v_j$ must be adjacent. Hence, $v_j$ must be one of $v_i$'s at most $k$ neighbors in $H$ that come after it.

Suppose now that $v_j$ is a type 3 vertex. Since the common neighbor $v_h$ has already been colored, we know that $i<h,j$, meaning that $v_h$ must be one of the at most $k$ clique neighbors of $v_i$ in $H$. Each of these neighbors can have at most $\Delta-1$ already-colored adjacent nodes different from $v_i$, for a total of at most $\gamma(\Delta-1)$ vertices at distance 2 from $v_i$. Moreover, since $v_h$ is adjacent to $v_i$, it must be one of $v_i$'s at most $k$ neighbors in $H$ that come after it. This concludes that $\alpha+\beta+\gamma$.

Let $x_{p,q}$ be the number of colors needed by Algorithm 1 to color $v_i$. Assume that $p = 2$ and $q = 1$. Each type 1 vertex forbids 3 colors to $v_i$. In type 2 vertices, only the node $v_j$ has already been colored, implying that only 1 color is forbidden to $v_i$. In type 3 vertices, instead, the node $v_h$ has already been colored and it can have at most $(\Delta - 1)$ already-colored neighbors. We conclude that:

$$x_{2,1} \leq 1 + 3\alpha + \beta + \gamma(\Delta - 1) \leq 1 + 3k + k(\Delta - 1) = 1 + k\Delta + 2k$$

When $p, q = 1$, each type 1 vertex forbids only 1 color to $v_i$, concluding that:

$$x_{1,1} \leq 1 + \alpha + \beta + \gamma(\Delta - 1) \leq 1 + 3k + k(\Delta - 1) = 1 + k\Delta$$

Finally, when $p = 0$ and $q = 1$ we can ignore type 1 vertices:

$$x_{0,1} \leq 1 + \beta + \gamma(\Delta - 1) \leq 1 + 3k + k(\Delta - 1) = 1 + k\Delta - k$$

□

**Corollary 2.** *Given any graph $G$ of treewidth $k$ it holds that:*

$$\lambda_{2,1} \leq k\Delta + 2k \qquad \lambda_{1,1} \leq k\Delta \qquad \lambda_{0,1} \leq k\Delta - k$$

# 3 Outerplanar graphs

We now consider outerplanar graphs. A graph is *outerplanar* if it has a planar embedding such that all vertices touch the exterior face. Clearly, any outerplanar graph is a planar graph.

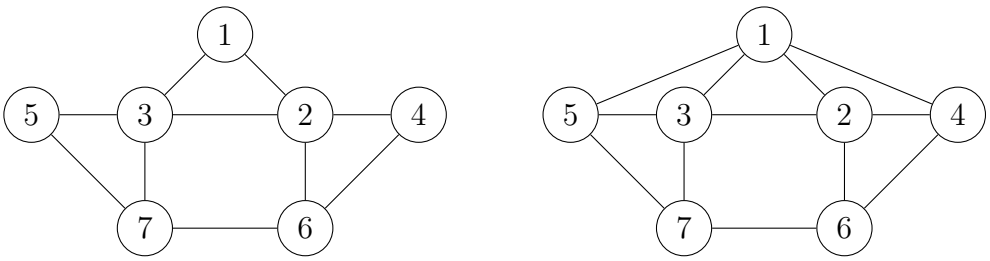



Figure 4: An outerplanar graph (left), a planar graph (right)

Upper bounds for this class of graphs are already known, in particular $\lambda_{2,1} \leq 2\Delta + 2$ was shown in [Jon93]. Using an algorithm similar to the previous one, we can improve this lower bound: we define a sequence of vertices and color them in reverse order. To define the sequence, the algorithm exploits the following lemma.

**Lemma 2.** *If $G$ is an outerplanar graph then one of the two holds:*

- *There exists a vertex of degree at most one.*
- *There exists a vertex of degree at most two which has a neighbor of degree at most four.*

```
1: function ALGORITHM 2(G, λ, p, q)
2:     Let H = G
3:     Let S = v_1, ..., v_n
4:     for i = 1, ..., n do
5:         Let u_i be a vertex that satisfies Lemma 2
6:         if u_i has two neighbors x, y that aren't adjacent in H then
7:             Add the edge (x, y) in H
8:         end if
9:         Set v_i = u_i
10:        Remove u_i from H
11:    end for
12:    for i = n, ..., 1 do
13:        Color v_i with the smallest available color in {0, ..., λ}
14: satisfying the L(p, q)-coloring constraints in G
15:    end for
16: end function
```

Figure 5: Algorithm for $\lambda$-Coloring outerplanar graphs

In particular, we notice that, while computing the sequence $v_1, \ldots, v_n$, the graph obtained by removing $u_i$ and adding the edge $(x, y)$ is still outerplanar, hence the lemma can always be applied. Using a counting argument similar to the one used for the previous algorithm, we obtain the following.

**Theorem 3.** *Given any outerplanar graph $G$, the Algorithm 2 finds:*

1. *An $L(2,1)$-labeling using the set $\{0, \ldots, \Delta + 8\}$.*
2. *An $L(1,1)$-labeling using the set $\{0, \ldots, \Delta + 4\}$.*
3. *An $L(0,1)$-labeling using the set $\{0, \ldots, \Delta + 2\}$.*

*Proof.* First, we notice that in each iteration $H$ is always outerplanar thanks to the edge (added or already existing) that connects the two neighbors of the removed node.

Assume that the sequence always picks a node $v_i$ with degree at most 2 who has a neighbor u of degree at most 4. Let $w$ be the other neighbor of $v_i$ (assuming it exists). We notice that $u$ and $w$ can have

at most $3+\Delta-1$ at distance 2 from $v_i$ that have already been colored. Moreover, $u$ and $w$ can also be colored already. Let $x_{p,q}$ be the number of colors needed by Algorithm 2 to color $v_i$. We get that:

$$x_{2,1} \leq 1 + 3 \cdot 2 + (3 + \Delta - 1) = 1 + \Delta + 8$$
$$x_{1,1} \leq 1 + 2 + (3 + \Delta - 1) = 1 + \Delta + 4$$
$$x_{0,1} \leq 1 + (3 + \Delta - 1) = 1 + \Delta + 2$$

□

**Corollary 3.** *Given any outerplanar graph $G$ it holds that:*

$$\lambda_{2,1} \leq \Delta + 8 \qquad \lambda_{1,1} \leq \Delta + 6 \qquad \lambda_{0,1} \leq \Delta + 2$$

# 4 Split graphs

We now move to the last class of graph, i.e. split graphs. A split graph is a graph $G$ whose node set can be partitioned into two sets $K$ and $S$ such that $K$ is a clique and $S$ is an independent set in $G$.

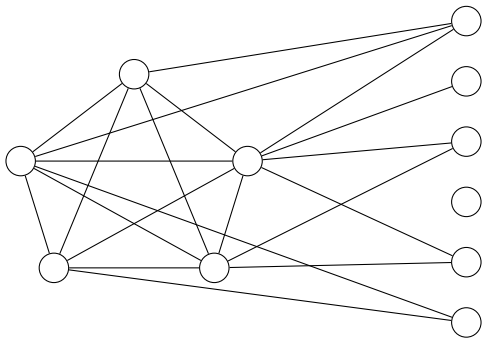


Figure 6: A split graph

Using an algorithm similar to the two previously shown ones, we can also find upper bounds for this class of graphs (see [BKT+04] for info on the third algorithm).

**Corollary 4.** *Given any split graph $G$ it holds that:*

$$\lambda_{2,1} \leq \frac{1}{2}\Delta^{1.5} + 2\Delta \qquad \lambda_{1,1} \leq \frac{1}{2}\Delta^{1.5} + \Delta \qquad \lambda_{0,1} \leq \frac{1}{2}\Delta^{1.5}$$

However, we won't focus on this third algorithm, as we are more interested in showing lower bounds for this class of graphs. In particular, we can show the following.

**Theorem 4.** *For any $\Delta > 0$, there is a split graph with maximum degree $\Delta$ such that:*

$$\lambda_{2,1} \geq \lambda_{1,1} \geq \lambda_{0,1} \geq \frac{1}{2}\sqrt{\frac{2}{3}}\Delta^{1.5}$$

*Proof.* Let $S$ be an independent set partitioned into $\sqrt{\frac{2}{3}\Delta}$ groups of $\frac{1}{3}\Delta$ nodes, for a total of $\frac{1}{2}\sqrt{\frac{2}{3}}\Delta^{1.5}$ nodes in $S$. We also notice that we have $\frac{\sqrt{\frac{2}{3}\Delta}\left(\sqrt{\frac{2}{3}\Delta}-1\right)}{2} \leq \frac{1}{3}\Delta$ distinct pairs of groups in this partition.

Let $K$ be a clique of $\frac{1}{3}\Delta + 1$ nodes. For each pair of groups, we take one unique node in the clique and make that node adjacent to each node in the two groups of the pair. Hence, each node in the clique now has at most $3 \cdot \frac{1}{3}\Delta = \Delta$ neighbors, making the graph a valid example. Moreover, we notice that this graph has diameter two and any pair of vertices in the independent set has distance exactly two. This means that we need at least one different color for each node in the independent set in order to satisfy the $L(0,1)$-constraints, meaning that we need at least $\frac{1}{2}\sqrt{\frac{2}{3}}\Delta^{1.5}$ nodes. □

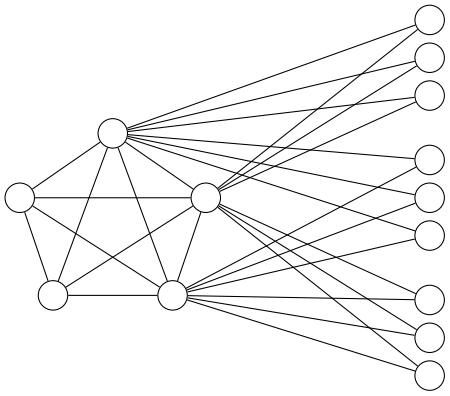


Figure 7: The graph constructed by the proof with $\Delta = 10$

This lower bound is almost equal to the upper bound, especially in the $\lambda_{0,1}$ case. We also notice that, by definition, each split graph is also a chordal graph. This implies that the lower bounds of the previous theorem also hold for chordal graphs, and thus also for $k$-trees.

**Corollary 5.** *For any $\Delta > 0$, there is a chordal graph with maximum degree $\Delta$ such that:*

$$\lambda_{2,1} \geq \lambda_{1,1} \geq \lambda_{0,1} \geq \frac{1}{2}\sqrt{\frac{2}{3}}\Delta^{1.5}$$

# References


[BB92] A.A. Bertossi and M.A. Bonuccelli. "Code assignment for hidden terminal interference avoidance in multihop packet radio networks". In: IEEE Computer Society Press, 1992. DOI: 10.1109/INFCOM.1992.263490.

[BKT+04] Hans L. Bodlaender, Ton Kloks, Richard B. Tan, et al. "Approximations for Lambda-Colorings of Graphs". In: *The Computer Journal* (2004). DOI: 10.1093/comjnl/47.2.193.

[GY92] Jerrold R. Griggs and Roger K. Yeh. "Labelling Graphs with a Condition at Distance 2". In: *SIAM Journal on Discrete Mathematics* (1992). DOI: 10.1137/0405048.

[HRS08] Frederic Havet, Bruce Reed, and Jean-Sebastien Sereni. "L(2,1)-labelling of graphs". In: *ACM-SIAM symposium on Discrete algorithms* (2008). URL: http://portal.acm.org/citation.cfm?id=1347082.1347151.

[Jon93] K. Jonas. "Graph Colorings Analogues with a Condition at Distance Two: L(2, 1)-labelings and list Lambda-labelings". PhD thesis. University of South Carolina, 1993.

[Nes08] Jaroslav Nesetril. "Structural Properties of Sparse Graphs". In: *Electronic Notes in Discrete Mathematics* (2008). DOI: https://doi.org/10.1016/j.endm.2008.06.050.

[Pat86] H. P. Patil. "On the structure of k-trees". In: *Journal of Combinatorics, Information and System Sciences,* (1986). URL: https://mathscinet.ams.org/mathscinet/relay-station?mr=0966069.

[Ros70] Donald J. Rose. "Triangulated graphs and the elimination process". In: *Journal of Mathematical Analysis and Applications* (1970). DOI: https://doi.org/10.1016/0022-247X(70)90282-9.