Approximations for λ-colorings of graphs

Simone Bianco - 1986936
Network Algorithms
Sapienza University of Rome

Introduction

Article: “Approximations for λ-colorings of graphs” by Bodlaender et al. (2004)

Focus: Upper and lower bounds for L(2,1), L(1,1) and L(0,1) labelings in various graph classes

Introduction

Article: “Approximations for λ-colorings of graphs” by Bodlaender et al. (2004)

Focus: Upper and lower bounds for L(2,1), L(1,1) and L(0,1) labelings in various graph classes

We will focus on Graphs with Treewidth k

Our objective

General upper bounds for L(2,1), L(1,1) and L(0,1) labelings are:

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

We will show that for any graph G of treewidth k it holds that:

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

Note: k is generally very small compared to Δ

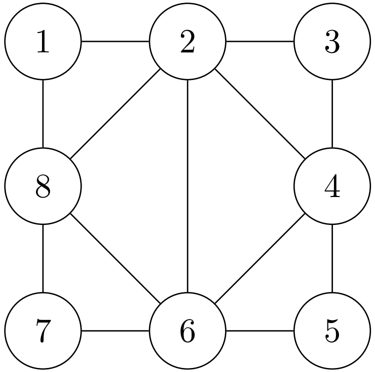
Tree decomposition

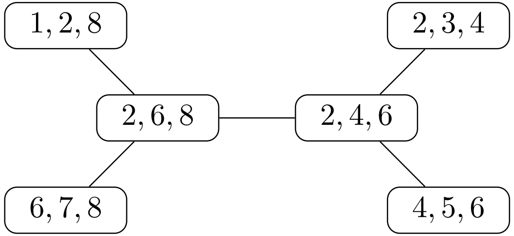
Given a graph G, a **tree decomposition** of G is a tree T whose vertices $X_1, \ldots, X_k$ are subsets of V(G) that satisfy the following properties:

1) $X_1, \ldots, X_k$ are a cover of V(G)

2) If $v \in X_i \cap X_j$ then each subset X_h in the path from X_i to X_j contains v

3) For each edge (u,v) of G at least one subset X_i contains u and v





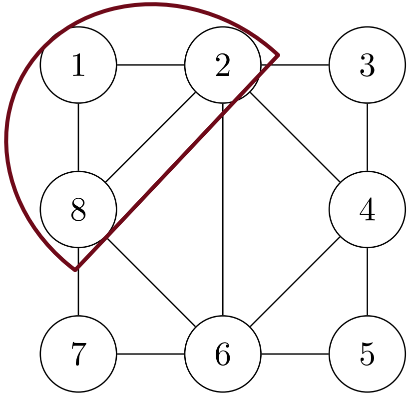
Tree decomposition

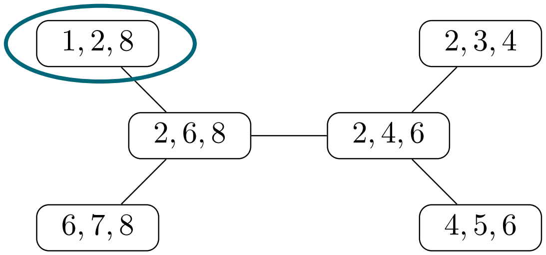
Given a graph G, a **tree decomposition** of G is a tree T whose vertices $X_1, \ldots, X_k$ are subsets of V(G) that satisfy the following properties:

1) $X_1, \ldots, X_k$ are a cover of V(G)

2) If $v \in X_i \cap X_j$ then each subset X_h in the path from X_i to X_j contains v

3) For each edge (u,v) of G at least one subset X_i contains u and v





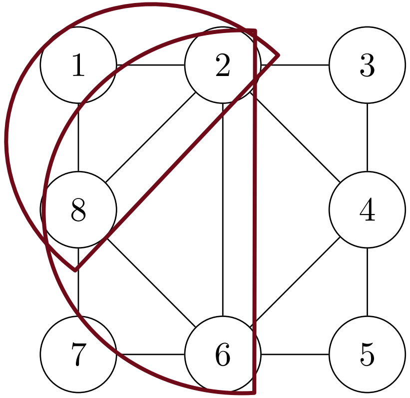
Tree decomposition

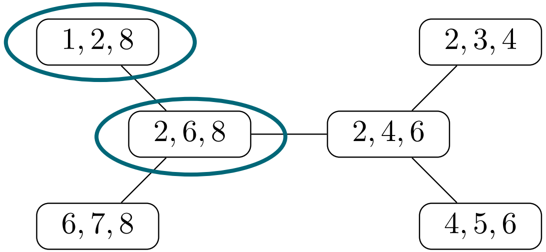
Given a graph G, a **tree decomposition** of G is a tree T whose vertices $X_1, \ldots, X_k$ are subsets of V(G) that satisfy the following properties:

1) $X_1, \ldots, X_k$ are a cover of V(G)

2) If $v \in X_i \cap X_j$ then each subset X_h in the path from X_i to X_j contains v

3) For each edge (u,v) of G at least one subset X_i contains u and v

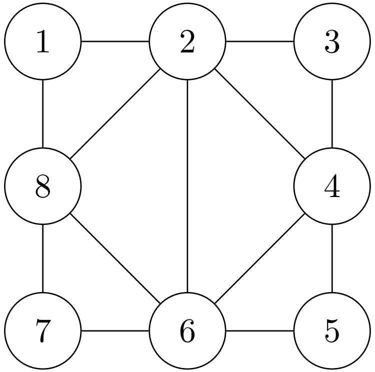


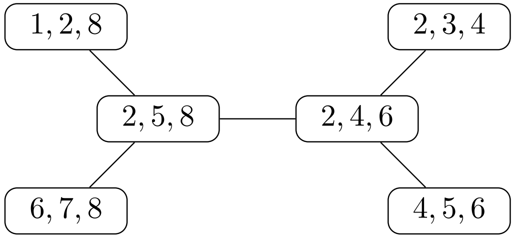


Treewidth

Width of a tree decomposition: size of the largest vertex of T, minus 1

Treewidth of a graph: smallest width of all its tree decompositions





Width = 2

The problem with Treewidth

Lots of graph problems that are NP-Hard can be solved in polynomial time if an optimal tree decomposition is known

The problem with Treewidth

Lots of graph problems that are NP-Hard can be solved in polynomial time if an optimal tree decomposition is known

If an upper bound on the treewidth of a graph is known, an optimal tree decomposition can be computed in linear time

Critical Problem: finding such upper bound is also NP-Hard!

The problem with Treewidth

Lots of graph problems that are NP-Hard can be solved in polynomial time if an optimal tree decomposition is known

If an upper bound on the treewidth of a graph is known, an optimal tree decomposition can be computed in linear time

Critical Problem: finding such upper bound is also NP-Hard!

The fastest way to get an upper bound on the treewidth of a graph is through **k-trees**

k-Trees

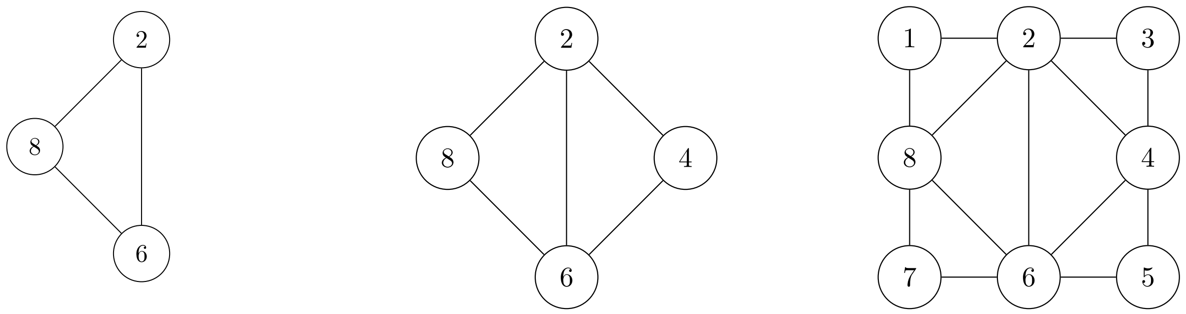
Inductive definition:

- The complete graph K_{k+1} is a k-tree
- A k-tree with $n > k+1$ nodes can be built from a k-tree G’ with n-1 nodes by adding a new node and connecting it to k vertices that form a k-clique in G’

k-Trees

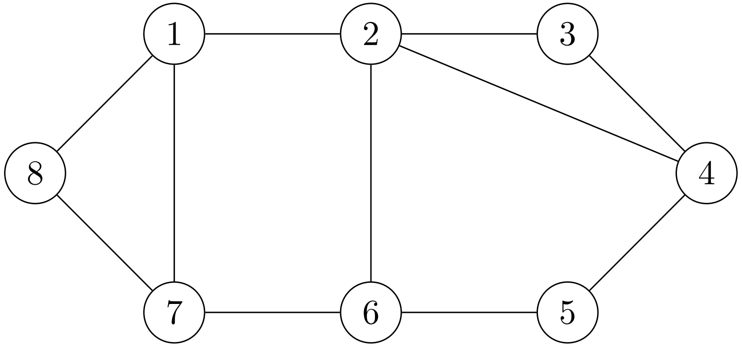
Inductive definition:

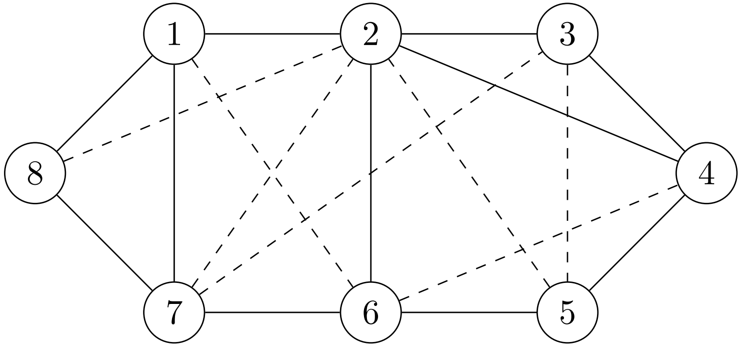
- The complete graph K_{k+1} is a k-tree
- A k-tree with $n > k+1$ nodes can be built from a k-tree G’ with n-1 nodes by adding a new node and connecting it to k vertices that form a k-clique in G’



Partial k-tree

Partial k-tree: any graph that is a subgraph of a k-tree





Treewidth and k-trees

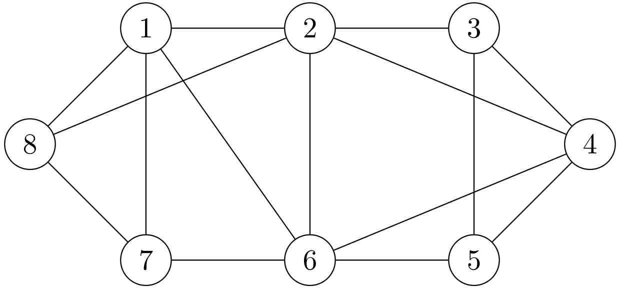
(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

(Cor) k is the smallest integer such that G is a partial k-tree if and only if $tw(G) \leq k$

We will use this fact to construct an algorithm based on k-trees

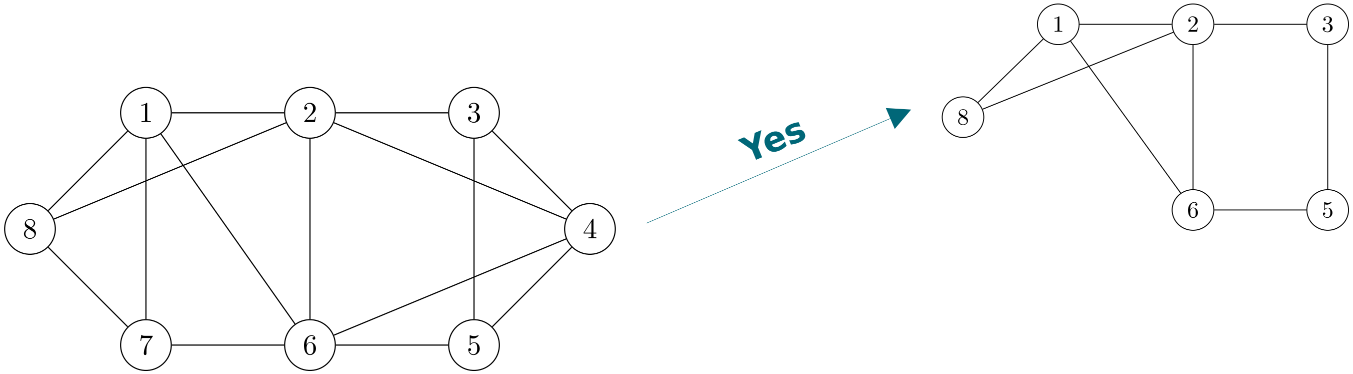
Induced subgraph

(Def.) Given a graph G = (V,E) and a subset of vertices S ⊆ V, the subgraph of G **induced** by S is the graph G[S] = (S, E’) such that (u,v) ∈ E’ if and only if u,v ∈ S



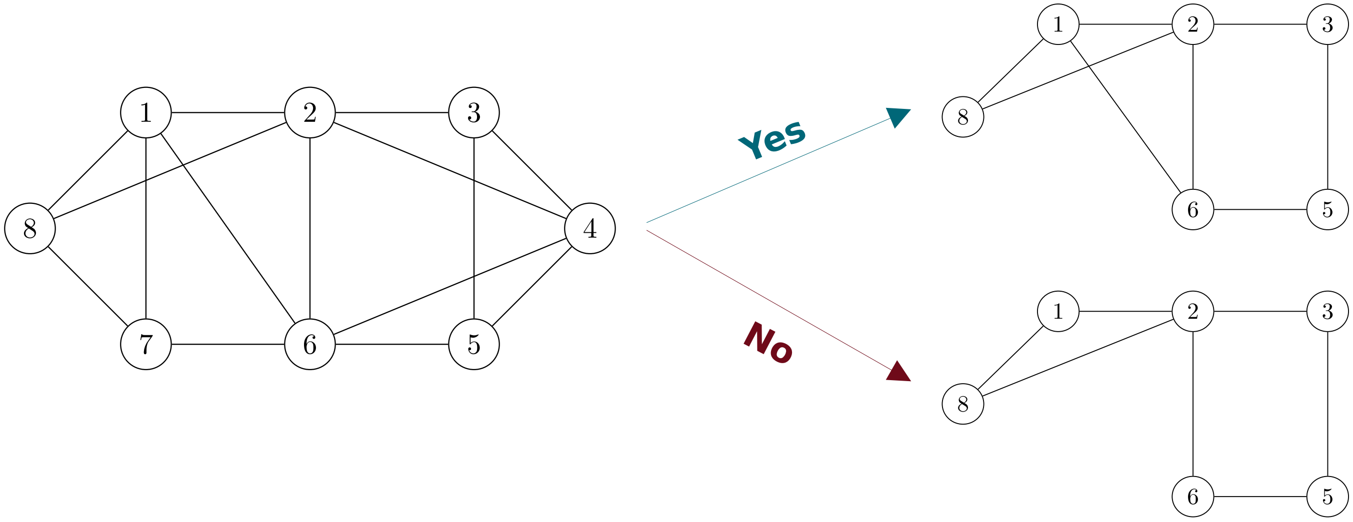
Induced subgraph

(Def.) Given a graph G = (V,E) and a subset of vertices S ⊆ V, the subgraph of G **induced** by S is the graph G[S] = (S, E’) such that (u,v) ∈ E’ if and only if u,v ∈ S



Induced subgraph

(Def.) Given a graph G = (V,E) and a subset of vertices S ⊆ V, the subgraph of G **induced** by S is the graph G[S] = (S, E’) such that (u,v) ∈ E’ if and only if u,v ∈ S



Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longrightarrow$) Let $G = (V,E)$ be a partial k-tree and let $H = (V, E')$ be the k-tree that contains G

Clearly $tw(G) \leq tw(H)$

We show by induction that $tw(H) \leq k$

Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longrightarrow$)

If $|V| = k+1$ then $H = K_{k+1}$

Hence $T = (\{V\}, \emptyset)$ is a tree decomposition of H of width k.

Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longrightarrow$)

If $|V| > k+1$ then there is a $v \in V$ such that:

- The induced subgraph $H' = H[V - \{v\}]$ is a k-tree
- $H[N(v)]$ has a k-clique $K = \{u_1, \ldots, u_k\}$ in H'

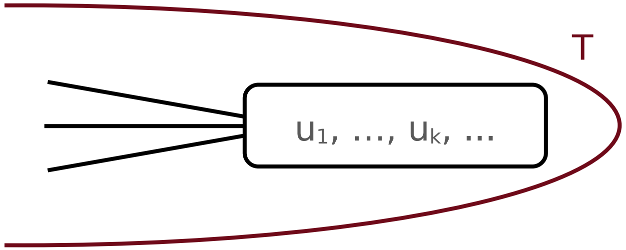
Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longrightarrow$)

H' has a tree decomp. $T = (\{X_1, \ldots, X_t\}, F)$ of width at most k

Since $K = \{u_1, \ldots, u_k\}$ is a k-clique in H', it must be contained inside some X_i



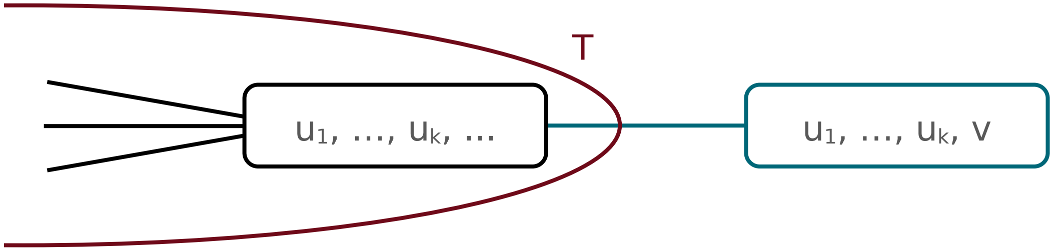
Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longrightarrow$)

Let $X_{t+1} = K \cup \{v\}$

$T' = (\{X_1, \dots, X_t, X_{t+1}\}, F \cup (X_i, X_{t+1}))$ is a tree decomp. of H that has width at most k



Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longleftarrow$) Let $T = (\{X_1, \ldots, X_t), F)$ be a tree decomp. of $G = (V,E)$ with width at most k

If $|V| = k+1$ then K_{k+1} is a k-tree that contains G

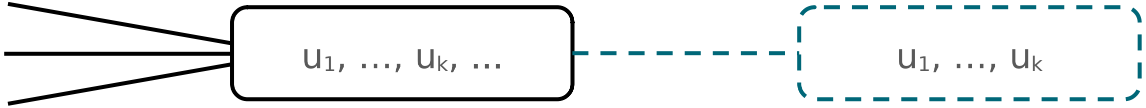
Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longleftarrow$)

If $|V| > k+1$, let X_i be a leaf of T and let X_j be its neighbor

We assume that $X_i \not\subseteq X_j$ since otherwise X_i can be removed

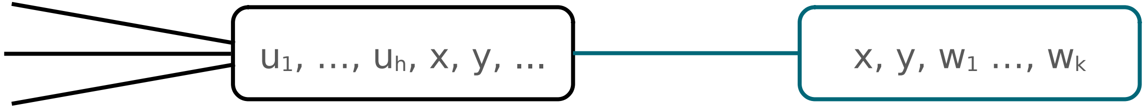


Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longleftarrow$)

We assume that $X_i \cap X_j = \{x\}$ since otherwise we can split X_i into multiple leaves in order to get a new decomposition for which the assumption holds

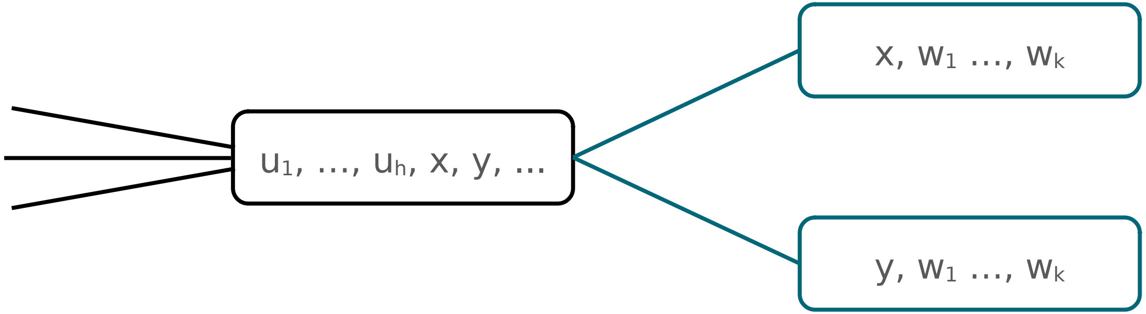


Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longleftarrow$)

We assume that $X_i \cap X_j = \{x\}$ since otherwise we can split X_i into multiple leaves in order to get a new decomposition for which the assumption holds



Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longleftarrow$)

Since T is has width at most k, X_i has at most k nodes

Moreover, all the nodes adjacent to v must be inside X_i

Treewidth and k-trees

(Thm) G is a partial k-tree if and only if $tw(G) \leq k$

$\Longleftarrow$)

Since T is has width at most k, X_i has at most k nodes

Moreover, all the nodes adjacent to v must be inside X_i

By inductive hypothesis, $G[V – \{v\}]$ is contained inside a k-tree H

By adding v to H and connecting it to the nodes in X_i we get a k-tree that contains G

■

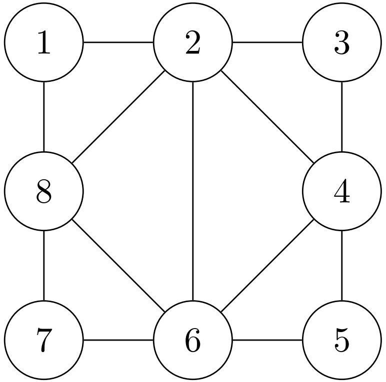
Chordal graphs

K-trees are a special type of **cordal (or triangulated)** graph.

Chordal graphs

K-trees are a special type of **cordal (or triangulated)** graph.

A graph is **chordal** when all cycles of 4+ vertices have a **chord** — an edge that is not part of the cycle but connects two vertices of the cycle.

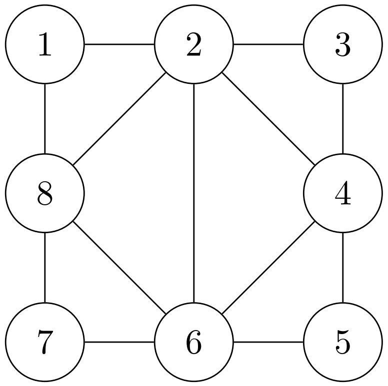


Chordal graphs

K-trees are a special type of **cordal (or triangulated)** graph.

A graph is **chordal** when all cycles of 4+ vertices have a **chord** — 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** (hence the alternative name).



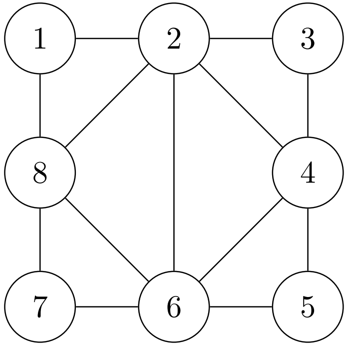
Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

A **perfect elimination ordering** is an ordering of the vertices such that for each node all of its neighbors that occur after it in the sequence form a clique with it.

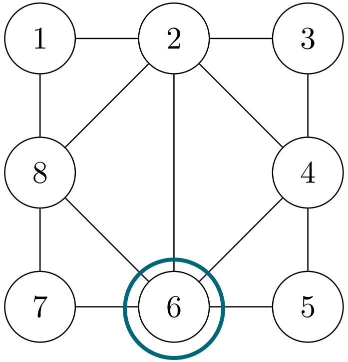


(1,3,5,7,4,2,8,6)

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

A **perfect elimination ordering** is an ordering of the vertices such that for each node all of its neighbors that occur after it in the sequence form a clique with it.

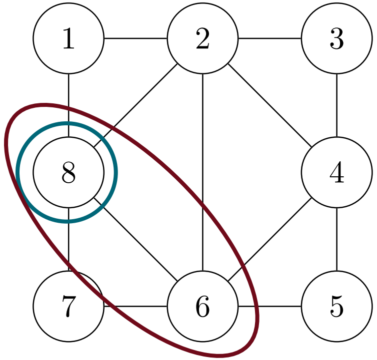


(1,3,5,7,4,2,8,6)

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

A **perfect elimination ordering** is an ordering of the vertices such that for each node all of its neighbors that occur after it in the sequence form a clique with it.

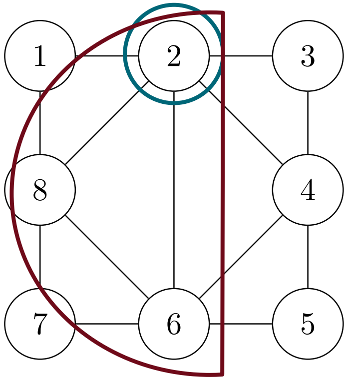


(1,3,5,7,4,2,8,6)

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

A **perfect elimination ordering** is an ordering of the vertices such that for each node all of its neighbors that occur after it in the sequence form a clique with it.

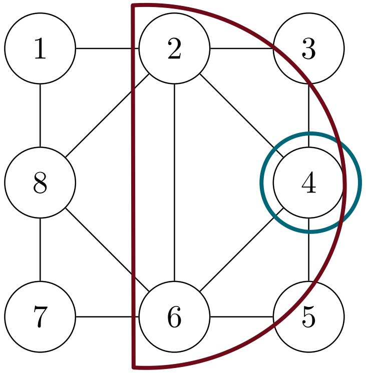


(1,3,5,7,4,2,8,6)

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

A **perfect elimination ordering** is an ordering of the vertices such that for each node all of its neighbors that occur after it in the sequence form a clique with it.

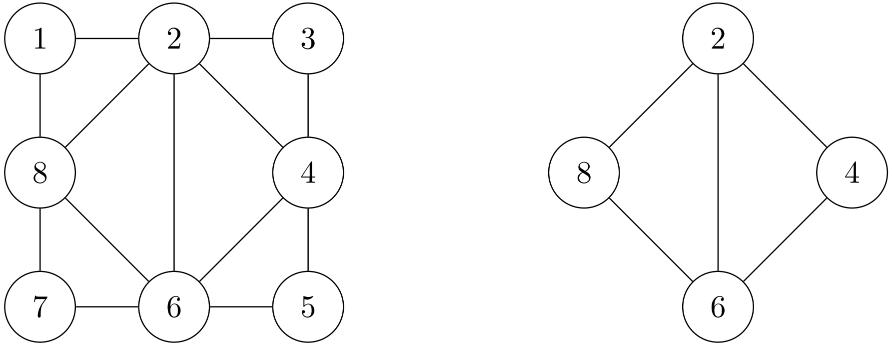


(1,3,5,7,4,2,8,6)

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

$\Longrightarrow$) **Claim:** G is chordal if and only if each induced subgraph of G is also chordal



Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

$\Longrightarrow$)

Let G = (V,E) be a chordal graph

If |V| = 1 then G is trivially chordal

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

$\Longrightarrow$)

If $|V| > 1$ then there is a vertex v who is part of a clique K

The graph $G[V - \{v\}]$ is chordal hence by inductive hypothesis it has a PEO $(v_1, \ldots, v_m)$

The ordering $(v, v_1, \ldots, v_m)$ is a PEO for G

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

$\Longleftarrow$) Let β be a PEO for G and let $C = \{v_1, \ldots, v_h\}$ be a cycle in G where $h > 3$

Let v_i be the first element of C in β

Perfect elimination ordering

(Thm) A graph is chordal if and only if it has a perfect elimination ordering

$\Longleftarrow$) Let β be a PEO for G and let $C = \{v_1, \dots, v_h\}$ be a cycle in G where $h > 3$

Let v_i be the first element of C in β

Since v_{i-1} and v_{i+1} come after v_i in β and $v_{i-1} \sim v_i \sim v_{i+1}$, they must form a clique in G

Hence, there is a chord separating C

∎

Algorithm for graphs with treewidth k

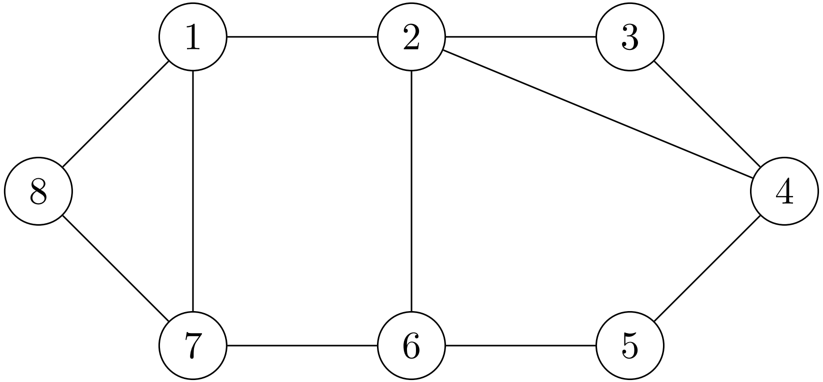
Given (G, λ, p, q) in input:

1) Build a k-tree H that contains G

2) Construct a PEO $v_1, \ldots, v_n$ on H

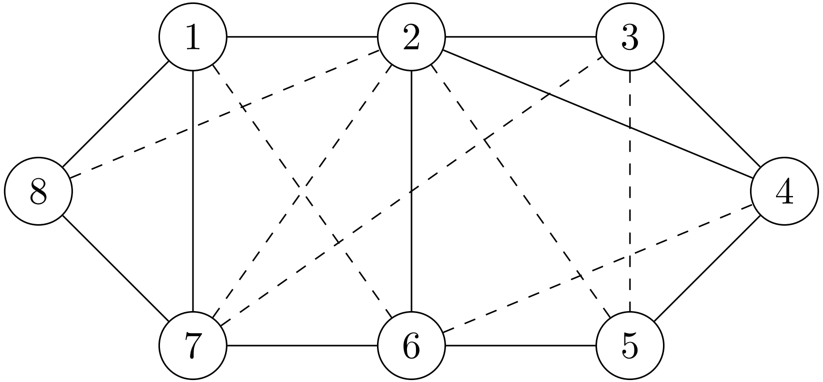
3) For i = n, …, 1:
   Color v_i using the smallest color in $\{0, \ldots, \lambda\}$ that satisfies the L(p,q)-constraints in G

Example of (2,1)-labeling



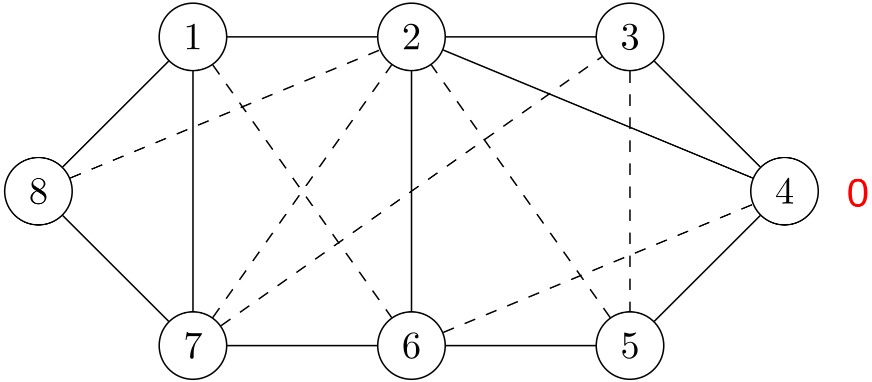
Treewidth: 3

Example of (2,1)-labeling



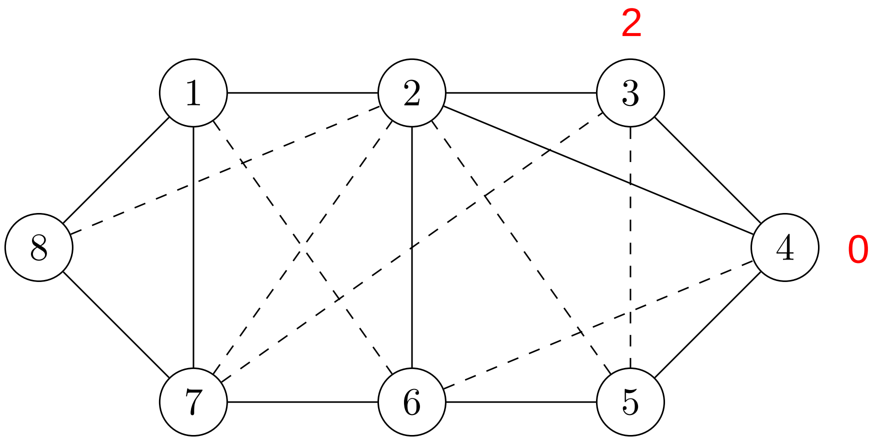
Sequence: (8,1,7,2,6,5,3,4)

Example of (2,1)-labeling



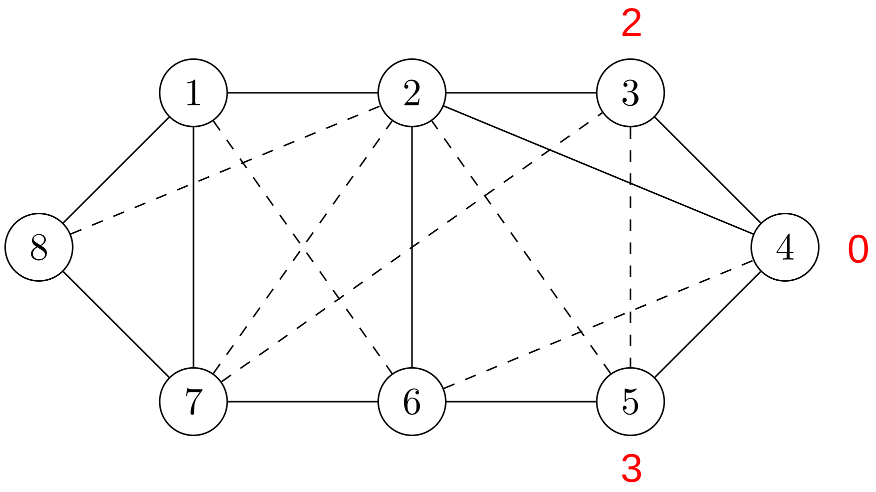
Sequence: (8,1,7,2,6,5,3,4) **Forbidden Colors: ---**

Example of (2,1)-labeling



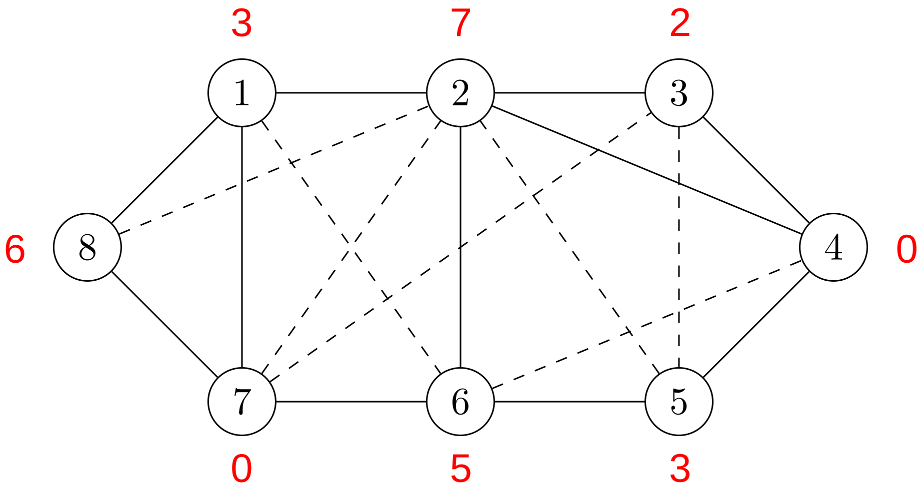
Sequence: (8,1,7,2,6,5,3,4) **Forbidden Colors: {0,1}**

Example of (2,1)-labeling



Sequence: (8,1,7,2,6,5,3,4) **Forbidden Colors: {0,1,2}**

Example of (2,1)-labeling



Sequence: (8,1,7,2,6,5,3,4) **λ ≥ 7 in order to work**

Algorithm for graphs with treewidth k

(Thm) Given any graph G of treewidth k, the previous algorithm can find:

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

(Cor) Given any graph G of treewidth k it holds that:

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

Algorithm for graphs with treewidth k

Dim. For each v_i there are 3 types of already-colored nodes that forbid colors to v_i:

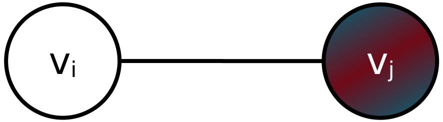
1) α vertices at distance 1 from v_i in G
2) β vertices at distance 2 from v_i in G that have a common neighbor with v_i in G that has not yet been colored
3) ɣ vertices at distance 2 from v_i in G that have a common neighbor with v_i in G that has already been colored

Algorithm for graphs with treewidth k

Dim. (cont.)

Let v_j be a node with $i < j$:

1) If v_j is a type 1 node than it is one of v_i's at most k clique-neighbors that are already colored

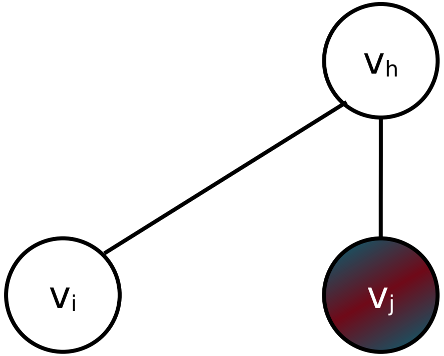


Algorithm for graphs with treewidth k

Dim. (cont.)

2) If v_j is a type 2 node and v_h is the common neighbor that has not yet been colored

$\Longrightarrow h < i < j$ and $v_i \sim v_h \sim v_j$



Algorithm for graphs with treewidth k

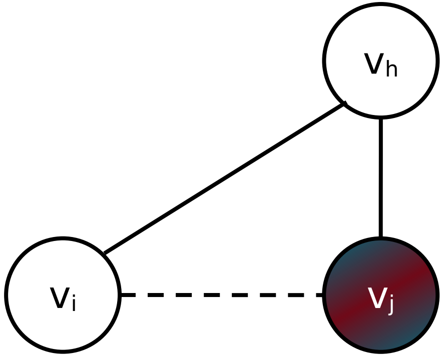
Dim. (cont.)

2) If v_j is a type 2 node and v_h is the common neighbor that has not yet been colored

   $\Longrightarrow h < i < j$ and $v_i \sim v_h \sim v_j$

   $\Longrightarrow v_i, v_j$ are in v_h's at most k clique-neighbors

   $\Longrightarrow v_j$ is one of v_i's at most k clique-neighbors that are already colored

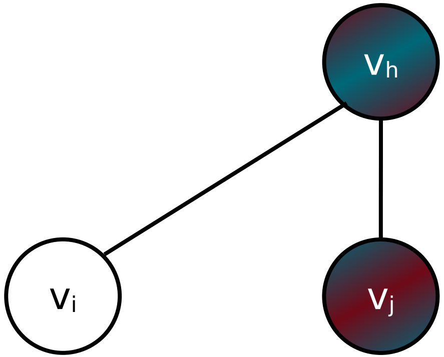


Algorithm for graphs with treewidth k

Dim. (cont.)

3) If v_j is a type 3 node and v_h is the common neighbor that has already been colored

$\Longrightarrow$ v_h is one of v_i's at most k clique-neighbors that are already colored



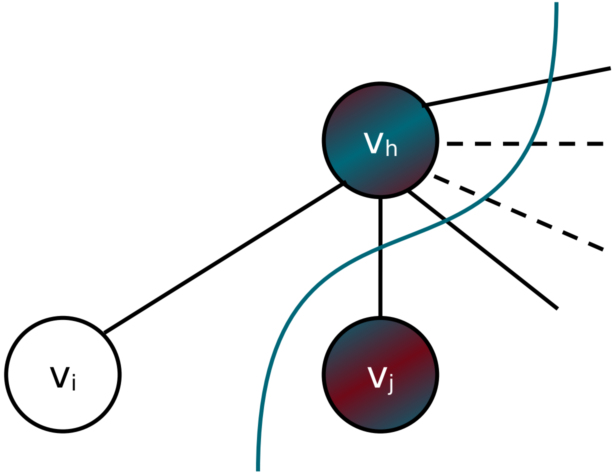
Algorithm for graphs with treewidth k

Dim. (cont.)

3) If v_j is a type 3 node and v_h is the common neighbor that has already been colored

$\Longrightarrow$ v_h is one of v_i's at most k clique-neighbors that are already colored

$\Longrightarrow$ $i < h$ hence v_h may have at most $\Delta - 1$ already colored neighbors (v_j included)



Algorithm for graphs with treewidth k

Dim. (cont.)

Each case has at least one of v_i's at most k clique-neighbors $\Longrightarrow \alpha + \beta + \gamma \leq k$

Algorithm for graphs with treewidth k

Dim. (cont.)

Each case has at least one of v_i's at most k clique-neighbors $\Longrightarrow \alpha + \beta + \gamma \leq k$

Let $x_{p,q}$ denote the number of colors needed to color v_i for L(p,q).

Algorithm for graphs with treewidth k

Dim. (cont.)

Each case has at least one of v_i's at most k clique-neighbors $\Longrightarrow \alpha + \beta + \gamma \leq k$

Let $x_{p,q}$ denote the number of colors needed to color v_i for L(p,q).

$X_{2,1} \leq 1 + 3\alpha + \beta + \gamma(\Delta\text{-}1) \leq 1 + k\Delta + 2k \quad \Longrightarrow \{0, \ldots, k\Delta + 2k\}$ suffices for L(2,1)

$X_{1,1} \leq 1 + \alpha + \beta + \gamma(\Delta\text{-}1) \leq 1 + k\Delta \quad \Longrightarrow \{0, \ldots, k\Delta\}$ suffices for L(1,1)

$X_{0,1} \leq 1 + \beta + \gamma(\Delta\text{-}1) \leq 1 + k\Delta - k \quad \Longrightarrow \{0, \ldots, k\Delta - k\}$ suffices for L(0,1)

■

Recap

We proved that for any graph G of treewidth k the following **upper bounds** hold:

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

However, the algorithm that we used requires time O(knΔ) which is **exponential**

Polynomial time algorithms are known for L(1,1) and L(0,1) labelings in graphs of treewidth k, but no algorithms are known for L(2,1) labelings

Main references

- Hans L. Bodlaender, Ton Kloks, Richard B. Tan, et al. “Approximations for Lambda-Colorings of Graphs”. In: The Computer Journal (2004)

- Jerrold R. Griggs and Roger K. Yeh. “Labelling Graphs with a Condition at Distance 2”. In: SIAM Journal on Discrete Mathematics (1992)

- Frederic Havet, Bruce Reed, and Jean-Sebastien Sereni. “L(2,1)-labelling of graphs”. In: ACM-SIAM symposium on Discrete algorithms (2008)

- H. P. Patil. “On the structure of k-trees”. In: Journal of Combinatorics, Information and System Sciences, (1986)

Thank you for the attention!