Testing superperfection of k-trees

T. Kloks, H. Bodlaender

RUU-CS-92-09
February 1992

Testing superperfection of k-trees

T. Kloks, H. Bodlaender

Technical Report RUU-CS-92-09
February 1992

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924-3275

Testing superperfection of k-trees

T. Kloks [*] H. Bodlaender [†]
Department of Computer Science, Utrecht University
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

An interval coloring of a weighted graph with non-negative weights, maps each vertex onto an open interval on the real line with width equal to the weight of the vertex, such that adjacent vertices are mapped to disjoint intervals. The total width of an interval coloring is defined as the width of the union of the intervals. The interval chromatic number of a weighted graph is the least total width of an interval coloring. The weight of a subset of vertices is the sum of the weights of the vertices in the subset. The clique number of a weighted graph is the weight of the heaviest clique in the graph. A graph is called superperfect if, for every non-negative weight function, the clique number is equal to the interval chromatic number.

A k-tree is a graph which can be recursively defined as follows. A clique with $k+1$ vertices is a k-tree. Given a k-tree with n vertices, a k-tree with $n+1$ vertices can be obtained by making a new vertex adjacent to all vertices of a k-clique in the k-tree.

In this paper we present, for each constant k, a linear time algorithm to test if a k-tree is superperfect. We also give, for each constant k, a constant time algorithm to produce a complete characterization of superperfect k-trees. Finally we present a complete list of critical non-superperfect 2-trees. Answering a question of Golumbic ([11]), this shows the existence of triangulated graphs which are superperfect but not comparability graphs.

1 Introduction

Since the discovery of perfect graphs in 1960, much research has been devoted to special classes of perfect graphs, such as comparability graphs and triangulated graphs. The class of triangulated graphs contains well known graph classes such as interval

[*]This author is supported by the foundation for Computer Science (S.I.O.N) of the Netherlands Organization for Scientific Research (N.W.O.), Email: ton@cs.ruu.nl.

[†]This author is partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075 (project ALCOM), Email: hansb@cs.ruu.nl.

graphs, split graphs, k-trees, and indifference graphs. The class of comparability graphs contains complements of interval graphs, permutation graphs, threshold graphs and P_4-free graphs (or cographs). Much work has been done in characterizing these graph classes and in finding relations between them. Interest has only increased since Lovász settled the perfect graph conjecture in 1972 ([13]). An explanation for this interest, from a theoretical point of view, might be the, as yet unsettled, strong perfect graph conjecture. From an algorithmic point of view, perfect graphs have become of great interest since the discovery of polynomial time algorithms (by Grötschel, Lovász and Schrijver) for NP-complete problems like Clique, Stable set and the Coloring problem, when restricted to perfect graphs (see [4]). From a practical point of view, special classes of perfect graphs have proven their importance by the large amount of applications (see for example [11] for applications in general and [6] for an overview of applications of interval graphs).

For computer science, the class of partial k-trees, which are subgraphs of k-trees, plays an increasingly important role. One reason for this is the existence of polynomial time algorithms for many NP-complete problems when restricted to partial k-trees for some constant k (see for example [2] and [1]). This has become even more interesting since the discovery of fast algorithms for the recognition of partial k-trees for constant k ([12], [3], [14], [17]). Of theoretical great importance is the work of Robertson and Seymour (see [18]), who showed that, for each k, there exists a finite set of forbidden minors for partial k-trees.

Most classes of perfect graphs can be recognized in polynomial time ([19], [5], [11], [16], [10]). An exception seems to be the class of superperfect graphs. Determining the interval chromatic number of a weighted interval graph with weights 1 and 2 is NP-complete. When restricted to weighted partial k-trees, for some constant k, and with weights bounded by some constant, it can be seen that the interval chromatic number can be determined in linear time, when the embedding of the graph in a k-tree is given. Until now we have not been able to find a polynomial algorithm to test superperfection on partial k-trees. In this paper we present our results for testing superperfection on k-trees.

The class of superperfect graphs contains that of the comparability graphs, but these classes are not equal as has been pointed out by Golumbic who showed the existence of an infinite class of superperfect graphs which are not comparability graphs (see [11]). However all these graphs are neither triangulated nor co-triangulated, and in [11] the question is therefore raised whether for triangulated graphs the classes of superperfect and comparability graphs coincide. For split graphs this equivalence has been shown. Our results show this is not the case in general. We show the existence of triangulated graphs which are superperfect but are not comparability graphs.

The results presented in this paper can be summarized as follows. We give a complete characterization, by means of forbidden induced subgraphs, of 2-trees which are superperfect. For each constant k we give a constant time algorithm which produces a complete characterization of superperfect k-trees, by means of forbidden configurations. With the aid of this characterization we find, for each constant k, a linear time algorithm to test superperfection of k-trees.

2 Preliminaries

We start with some definitions and easy lemmas. Most definitions and results in this section are taken from [11]. For further information on perfect graphs the reader is referred to this book or to [4].

Definition 2.1
An undirected graph $G = (V, E)$ is called a comparability graph, or a transitively orientable graph, if there exists an orientation (V, F) satisfying

$$F \cap F^{-1} = \emptyset \wedge F + F^{-1} = E \wedge F^2 \subseteq F$$

Such an orientation is called a *transitive* orientation.

So if F is a transitive orientation then $(a, b) \in F$ and $(b, c) \in F$ imply $(a, c) \in F$.

It is easily checked that if a graph G is a comparability graph, then this also holds for every induced subgraph of G. A comparability graph can not contain an induced odd cycle of length at least 5, or the induced complement of a cycle with length at least 5. This last part can be seen as follows: consider the complement of a cycle with length at least 6 and assume there is a transitive orientation. Since the orientation is acyclic, there must exist at least one sink node s. Consider the square which contains exactly two neighbors of s but not s itself. The subgraph induced by the square and s can not be transitively oriented such that s is a sink node. In [11] it is shown that comparability graphs are *perfect* (i.e. for every induced subgraph the chromatic number is equal to the maximum size of a clique) and can be recognized in polynomial time (see also [19]). Comparability graphs share all these properties with *triangulated* graphs.

Definition 2.2
A graph is triangulated if it contains no chordless cycle of length greater than three.

Definition 2.3
Let $G = (V, E)$ be a graph. A *simplicial vertex* of G is a vertex of which the neighborhood forms a clique. An ordering of the vertices $\sigma = [v_1, \ldots, v_n]$ is called a *perfect elimination scheme* if each v_i is a simplicial vertex in $G[\{v_i, \ldots, v_n\}]$, which is the subgraph induced by $\{v_i, \ldots, v_n\}$.

Fulkerson and Gross ([8]) characterized triangulated graphs by means of a perfect elimination scheme.

Lemma 2.1 *A graph is triangulated if and only if there exists a perfect elimination scheme.*

A special type of triangulated graphs are *k-trees*.

Definition 2.4
A k-tree is defined recursively as follows: A clique with $k+1$ vertices is a k-tree; given a k-tree T_n with n vertices, a k-tree with $n+1$ vertices is constructed by making a new vertex x_{n+1} adjacent to a k-clique of T_n, and nonadjacent to the $n-k$ other vertices of T_n.

A triangulated graph is a k-tree if it is connected, every maximal clique contains $k+1$ vertices and every minimal vertex separator is a k-clique. It is clear that for k-trees there exists a perfect elimination scheme $\sigma = [v_1, \ldots, v_n]$ such that for each $1 \leq i \leq n-k$, the neighborhood of v_i is a clique with k vertices in the subgraph induced by $\{v_{i+1}, \ldots, v_n\}$. Notice that for $k=1$, k-trees are just ordinary trees. So k-trees are a natural generalization of trees.

A *weighted* graph is a pair (G, w), where G is a graph and w is a weight function which associates to every vertex x a non-negative weight $w(x)$. For a subset S of the vertices we define the weight of S, $w(S)$ as the sum of the weights of the vertices in S.

Definition 2.5
An *interval coloring* of a weighted graph (G, w) maps each vertex x to an open interval I_x on the real line, of width $w(x)$, such that adjacent vertices are mapped to disjoint intervals. The *total width* of an interval coloring is defined to be $|\bigcup_x I_x|$. The *interval chromatic number* $\chi(G, w)$ is the least total width needed to color the vertices with intervals.

Determining whether $\chi(G, w) \leq r$ is an NP-complete problem, even if w is restricted to values 1 and 2 and G is an interval graph, this has been shown by L. Stockmeyer as reported in [11]. In this paper we shall only use the following alternative definition of the interval chromatic number (see [11]).

Theorem 2.1 *If (G, w) is a weighted undirected graph, then*

$$\chi(G, w) = \min_F(\max_\mu w(\mu))$$

where F is an acyclic orientation of G and μ is a path in F.

If w is a weighting and F is an acyclic orientation, then we say that F is a *superperfect orientation* with respect to w if the weight of the heaviest path in F does not exceed the weight of the heaviest clique.

Definition 2.6
The *clique number* $\Omega(G, w)$ of a weighted graph (G, w) is defined as the maximum weight of a clique in G.

We use the capital Ω to avoid confusion with the weighting w. It is easy to see that $\Omega(G, w) \leq \chi(G, w)$ holds for all weighted graphs, since for any acyclic orientation and for every clique there exists a path in the orientation which contains all vertices of the clique.

Definition 2.7
A graph G is called *superperfect* if for *every non-negative weight function* w, $\Omega(G, w) = \chi(G, w)$.

Notice that each induced subgraph of a superperfect graph is itself superperfect, and also that every superperfect graph is perfect. If G is a comparability graph, then there exists an orientation such that every path is contained in a clique. This proves the following theorem (see also [11]).

Theorem 2.2 *A comparability graph is superperfect.*

The converse of this theorem is not true. In [11] an infinite class of superperfect graphs is given that are not comparability graphs. However, none of these graphs is triangulated. In [11] (page 214) the question is raised if the converse of the theorem holds for triangulated graphs; is it true or false that, for *triangulated* graphs, G is a comparability graphs if and only if G is superperfect? In the next section we answer this question in the negative, and we give a complete characterization of superperfect 2-trees.

3 2-trees and superperfection

In this section we give a characterization of 2-trees that are superperfect by means of forbidden subgraphs. In 1967 Gallai, [9], published a complete list of critical non-comparability graphs (this list can also be found in [4] page 78). Extracting from this list the triangulated graphs which are subgraphs of 2-trees (or: have *treewidth* at most two), we find a characterization of 2-trees which are comparability graphs. We find two types of forbidden induced subgraphs, which we call the 2-*star* and the *odd wing*. They are illustrated in figure 1. Notice that a 2-star and a wing are 2-trees, and that a wing has at least seven vertices. We call a wing odd (even) if the total number of vertices is odd (even). The following lemma is easy to check.

Lemma 3.1 *A wing is a comparability graph if and only if it is even.*

We thus find the following characterization of 2-trees that are comparability graphs.

Theorem 3.1 *A 2-tree is a comparability graph if and only if it does not contain a 2-star or an odd wing.*

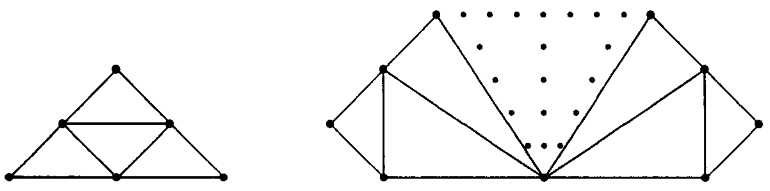

Figure 1: 2-star (left) and wing (right)

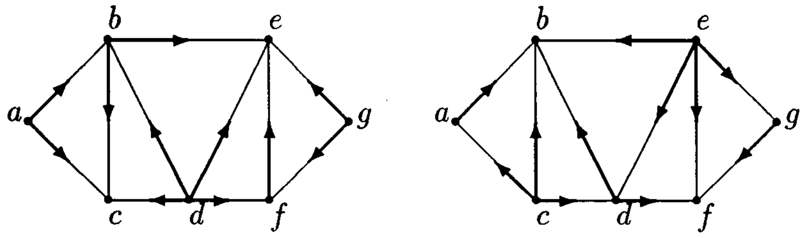

Figure 2: two orientations of the wing with seven vertices

The next theorem shows that the smallest odd wing, with seven vertices, (which is not a comparability graph) is superperfect. As we shall see later, this is in fact the only odd wing that is superperfect.
Remark. Notice that in [11] (page 212, figure 9.9) this graph is mistakenly placed in the position of a non-superperfect graph. See also [15] and [7]; the result of [15] is wrong: A wing is an interval graph.

Theorem 3.2 *The odd wing with seven vertices is superperfect.*

Proof:
Label the vertices of the graph as in figure 2. We consider two orientations of this wing as illustrated in figure 2 and we show that for every weighting one of these orientations is superperfect. Notice that both orientations are such that there is exactly one path not contained in a triangle. In the first orientation this is the path $\{a, b, e\}$ and in the second orientation the path $\{c, d, f\}$. Consider a non-negative weighting w of the vertices. Suppose the orientation of the first type is *not* superperfect with respect to w. Then the path $\{a, b, e\}$ must be heavier then every triangle. Since $\{a, b, c\}$ is a triangle, this implies that $w(e) > w(c)$. But then $w(\{c, d, f\}) < w(\{e, d, f\})$, and since $\{e, d, f\}$ is a triangle, the second orientation is superperfect with respect to w. □

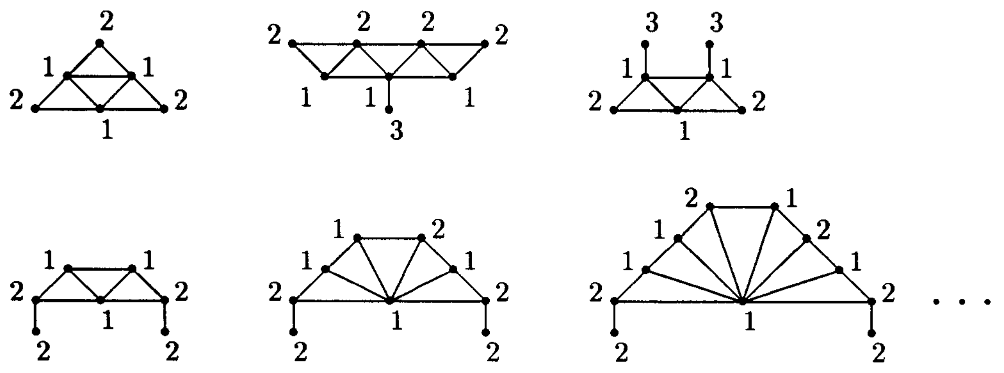

Figure 3: critical non-superperfect graphs

In the last part of this section we give a complete characterization of the superperfect 2-trees. In figure 3 we give an (infinite) list of forbidden induced subgraphs. The following lemma can be easily checked.

Lemma 3.2 *The graphs illustrated in figure 3 are not superperfect. The weight function that is shown is such that for any acyclic orientation, there exists a path which is heavier than the heaviest clique.*

Theorem 3.3 *A 2-tree is superperfect if and only if it does not contain an induced subgraph isomorphic to one of the graphs shown in figure 3.*

Proof:
Assume the 2-tree G has no induced subgraph from this list. Then the graph can not have an induced odd wing with 9 or more vertices. We may assume the graph is not a comparability graph, hence it contains an odd wing with 7 vertices. Consider the labeled wing of figure 2. Let H be the subgraph obtained from this wing by removing the vertices a and g. Let (x, y) be an edge of H. We say that C is a component at (x, y) if C is a maximal connected subgraph of $G[V \setminus \{x, y\}]$ *containing no vertices of* H. If C is a component at (x, y), we say that x is a degenerate vertex of this component if x is adjacent to all vertices of C. Notice that there are only four edges of H at which there can be components, namely (b, c), (b, d), (d, e) and (e, f), otherwise the 2-star would be an induced subgraph. The following remarks restrict the possible components.

1. If C is a component at (x, y), then either x is degenerate or y is degenerate.
2. For components at (b, d), b must be degenerate and for components at (d, e), e must be degenerate, otherwise the second graph in the list is an induced subgraph.

3. Consider components at (b, c) with at least two vertices. Either b or c is degenerate for *all* these components, otherwise the second graph from list 3 is an induced subgraph.

4. If there is a component at (b, c) with at least two vertices for which c is degenerate, then for all components at (e, f), e must be degenerate, otherwise the fourth graph in the list is an induced subgraph. If there is a component at (b, c) with at least two vertices for which b is degenerate then for all components at (e, f), f must be degenerate, otherwise the third subgraph is an induced subgraph. Without loss of generality, we assume that for all components at (b, c), c is degenerate and for all components at (e, f), e is degenerate.

5. If there is a component at (b, d) and one at (d, e), then the third graph in the list is an induced subgraph, hence this can not be the case.

6. If there is a component at (e, f) with at least two vertices (for which e is degenerate) then there can be no component at (b, d), otherwise the third graph from the list is an induced subgraph.

7. If there is a component at (b, c) with at least two vertices, then all components at (b, d) can have only one vertex, otherwise the second graph of the list is an induced subgraph.

8. Suppose there is a component at (b, d). Then all components at (b, c) can have at most two vertices. Furthermore, if there is a component at (b, c) with two vertices, then it is the only component at (b, c). Otherwise, the third subgraph of the list is an induced subgraph.

It follows that only one of two cases can occur.

- There is a component at (b, d). Then there is no component at (d, e). All components at (b, d) and at (e, f) have one vertex. Either all components at (b, c) have one vertex or there is only one component at (b, c), in which case it can have at most two vertices and c is degenerate.

- There are only components at (b, c), (d, e) and (e, f). For all components at (b, c), c is degenerate. For all components at (d, e) and at (e, f), e is degenerate.

It is easily checked (e.g. by methods described in the next section), that both types are superperfect. □

Notice that the list of forbidden induced subgraphs is infinite. In the next section we show that for each k there is a finite characterization of superperfect k-trees, by means of forbidden configurations. Furthermore we give for each k a constant time algorithm to find this characterization. As a consequence we find a linear time algorithm to check if a k-tree is superperfect.

4 k-trees and superperfection

In this section, let k be some constant, and let G be a k-tree. We start by showing that we can restrict the set of orientations to test if G is superperfect.

Definition 4.1
A *coloring* of a triangulated graph $G(V, E)$ with $k+1$ colors is a function $C : V \rightarrow \{1, \ldots, k+1\}$, such that $C(x) \neq C(y)$ whenever x and y are adjacent.

In this paper we only use colorings with $k+1$ colors; we do not always mention the number of colors. Notice that a coloring of a k-tree is unique up to a permutation of the colors:

Lemma 4.1 *If C and C' are two colorings of a k-tree $G = (V, E)$ then there exists a permutation π of the colors $\{1, \ldots, k+1\}$ such that for every vertex x, $C(x) = \pi(C'(x))$.*

Since a coloring of a k-tree is unique up to a permutation of the colors, the following set of orientations is uniquely defined for each k-tree.

Definition 4.2
Let G be a graph and let C be a coloring of G with $k+1$ colors. For each permutation π of the colors we define an orientation F_π as follows. Direct the edge (x, y) from x to y if $\pi(C(x)) < \pi(C(y))$. We define $\mathcal{F}_c(G)$ as the set of orientations obtained in this way.

The following lemma follows immediately from definition 4.2.

Lemma 4.2 *If G is a k-tree then:*

1. $|\mathcal{F}_c| = (k+1)!$.
2. *Each $F \in \mathcal{F}_c$ is acyclic.*
3. *If $F \in \mathcal{F}_c$ then each path μ in F has at most $k+1$ vertices.*

Definition 4.3
Let G be a k-tree. We define $\mathcal{F}^*$ as the set of acyclic orientations of G, of which every path contains at most $k+1$ vertices.

Notice that $\mathcal{F}_c \subset \mathcal{F}^*$.

Lemma 4.3 $\mathcal{F}_c = \mathcal{F}^*$.

Proof:
We proof that $|\mathcal{F}^*| = (k+1)!$. Let $F \in \mathcal{F}^*$. Let S be a k-clique in G, and let x and y be two vertices which are adjacent to all vertices of S. Since S is a clique and F is acyclic, there is a unique ordering of the vertices of S, say $s_1, s_2, \ldots, s_k$, such that $s_i \rightarrow s_j$ if and only if $i > j$. Since x is adjacent to all vertices of S, there exists an index $0 \leq t_x \leq k$ such that $x \rightarrow s_i$ for all $1 \leq i \leq t_x$ and $s_j \rightarrow x$ for all $t_x < j \leq k$. The same holds for y with index t_y. Consider the case $t_x < t_y$. Then F has a path of length $k+2$:

$$(s_k, s_{k-1}, \ldots, s_{t_y+1}, y, s_{t_y}, \ldots, s_{t_x+1}, x, s_{t_x}, \ldots, s_1)$$

Since $F \in \mathcal{F}^*$, we find that $t_x = t_y$. Now consider the recursive construction of G as a k-tree. Start with an acyclic orientation of a $(k+1)$-clique. This can be done in $(k+1)!$ manners. If we add a new vertex v and make it adjacent to a k-clique, by the argument above, the orientations of the edges incident with v are determined. Hence $|\mathcal{F}^*| = (k+1)!$. □

Definition 4.4
Let F be an acyclic orientation. A path μ in F is *contained* in a path μ', if all vertices of μ are also vertices of μ'.

Lemma 4.4 *Let $F \in \mathcal{F}_c$. Then any path μ in F is contained in a path μ' with $k+1$ vertices.*

Proof:
Let C be a coloring and let $F = F_\pi$. The colors in the path μ must appear in the same order as in the permutation. Assume there is a gap between adjacent colors c_1 and c_2 in the path (i.e. there is a color in the permutation between c_1 and c_2). Since the edge of the path with colors c_1 and c_2 is contained in a $(k+1)$-clique, the missing colors can be put between c_1 and c_2. Thus we can make a longer path μ' containing μ. □

Theorem 4.1 *Let G be a k-tree. Then G is superperfect if and only if*

$$\forall_w [\min_{F \in \mathcal{F}_c} \max_\mu w(\mu) = \Omega(G, w)]$$

Proof:
Assume G is superperfect. Let w be a non-negative weighting. There is an orientation F, such that $\max_\mu w(\mu) = \Omega(G, w)$. If every path in F has at most $k+1$ vertices, we are done. Assume F has a path with more than $k+1$ vertices. Now increase all weights with some constant $L > \Omega(G, w)$. Let w' be this new weighting. Notice that $\Omega(G, w') = \Omega(G, w) + (k+1)L$. Since G is superperfect, there must

be an orientation F' for this new weighting w'. Suppose F' also has a path μ with more than $k+1$ vertices. Then (with $|\mu|$ the number of vertices of μ):

$$\begin{aligned}\Omega(G,w') = \Omega(G,w) + (k+1)L &\geq w'(\mu) \\ &= w(\mu) + |\mu|L \\ &\geq w(\mu) + (k+2)L \\ &\geq (k+2)L\end{aligned}$$

Since $L > \Omega(G,w)$, this is a contradiction. We may conclude that $F' \in \mathcal{F}^* = \mathcal{F}_c$. We show that F' is also a good orientation for the weighting w. Let ν be a path in F'. By lemma 4.4, ν is contained in a path ν^* with $k+1$ vertices. Hence

$$w(\nu) \leq w(\nu^*) = w'(\nu^*) - (k+1)L \leq \Omega(G,w') - (k+1)L = \Omega(G,w)$$

The converse is trivial. □

Definition 4.5
Let G be a triangulated graph and let C be a coloring of G with $k+1$ colors. For each permutation π of the colors, let $\mathcal{P}(\pi)$ be the set of paths in F_π, which are *not* contained in a clique, and which have $k+1$ vertices. If $\mathcal{Q}$ is a set of paths in G, we say that $\mathcal{Q}$ forms a *cover* if, for every permutation π, there is a path $\mu \in \mathcal{Q}$ which can be oriented such that it is in $\mathcal{P}(\pi)$. A cover is called *minimal* if it contains $\frac{1}{2}(k+1)!$ paths.

Lemma 4.5 *Let G be a k-tree. If for some permutation π, $\mathcal{P}(\pi) = \emptyset$, then G is a comparability graph (hence superperfect).*

Proof:
Suppose $\mathcal{P}(\pi) = \emptyset$. Consider the orientation F_π. If there is a path in F_π which is not contained in a clique then, by lemma 4.4, $\mathcal{P}(\pi)$ can not be empty. Hence, every path in F_π is contained in a clique. Since F_π is acyclic, the lemma follows. □

Definition 4.6
Let G be a k-tree and let C be a coloring of G. Let S be a maximal clique of G, and let $\mathcal{Q}$ be a minimal cover. Define $LP(G,S,\mathcal{Q})$ as the following set of inequalities:

1. For each vertex x: $w(x) \geq 0$.
2. For each maximal clique $S' \neq S$: $w(S') \leq w(S)$.
3. For each path μ of $\mathcal{Q}$: $w(\mu) > w(S)$.

We call the second type of inequalities, the *clique inequalities*. The inequalities of the third type are called the *path inequalities*.

Lemma 4.6 *There are $n-k-1$ clique inequalities and $\frac{1}{2}(k+1)!$ path inequalities.*

Proof:
Notice that a k-tree has $n-k$ cliques with $k+1$ vertices. $\square$

Theorem 4.2 *Let G be a k-tree with a coloring C. G is* not *superperfect if and only if there is a maximal clique S and a minimal cover $\mathcal{Q}$ such that $LP(G, S, \mathcal{Q})$ has a solution.*

Proof:
Suppose $LP(G, S, \mathcal{Q})$ has a solution. Take this solution as a weighting. Then, clearly, for any orientation F_π there is a path (in $\mathcal{Q}$ and in $\mathcal{P}(\pi)$) which is heavier than the heaviest clique S. By theorem 4.1 G is not superperfect. On the other hand, if G is not superperfect, there exists a weighting w such that for every orientation F_π there is a path which is heavier than the heaviest clique (hence it can not be contained in a clique). By lemma 4.4 we may assume this path has $k+1$ vertices, hence it is in $\mathcal{P}(\pi)$. Take S to be the heaviest clique and let $\mathcal{Q}$ be a minimal cover for these paths. $\square$

Notice that we could use theorem 4.2 to make a polynomial time algorithm to test superperfection on k-trees: There are at most n^{k+1} different paths of length k, hence the number of minimal covers is at most $(n^{k+1})^{\frac{1}{2}(k+1)!}$. Since the number of maximal cliques of G is at most $n-k$, we only have to check for a polynomial number of sets of inequalities if it has a solution. This checking can be done in polynomial time, e.g. by the ellipsoid method. We now show that there also exists a linear time algorithm.

Consider a set of inequalities $LP(G, S, \mathcal{Q})$ which has a solution. Notice that if some vertex y does *not* appear in the path inequalities then we can set the weight $w(y) = 0$. This new weighting is also a solution. Hence we can transform the set of inequalities as follows:

Definition 4.7
Let G be a k-tree and let C be a coloring of G. Let S be a maximal clique, and let $\mathcal{Q}$ be a cover. Let H be the subgraph of G induced by the vertices of S and of all paths in $\mathcal{Q}$. Define $LP'(H, S, \mathcal{Q})$ as the following set of inequalities:

1. For each vertex x of H: $w(x) \geq 0$.
2. For each maximal clique $S' \neq S$ of H: $w(S') \leq w(S)$.
3. For each path μ in $\mathcal{Q}$: $w(\mu) > w(S)$.

The following lemma follows directly from the definitions 4.6 and 4.7.

Lemma 4.7 *$LP(G, S, \mathcal{Q})$ has a solution if and only if $LP'(H, S, \mathcal{Q})$ has a solution.*

Lemma 4.8 *The number of inequalities of $LP'(H,S,Q)$ is bounded by a constant.*

Proof:
There are $\frac{1}{2}(k+1)!$ paths in Q, each involving $k+1$ variables. The clique S has also $k+1$ vertices, hence it follows that the subgraph H, has at most $(\frac{1}{2}(k+1)!+1)(k+1)$ vertices. Since H is triangulated, the number of maximal cliques in H is bounded by the number of vertices. Hence the number of clique inequalities is bounded by $(\frac{1}{2}(k+1)!+1)(k+1)$. □

Notice that we now have the following algorithm to test superperfection of k-trees.
Algorithm to check superperfection of G

- Generate all $(k+1)$-colored triangulated graphs H, with at most $(1+\frac{1}{2}(k+1)!)(k+1)$ vertices, for which there exists:

  1. A maximal clique S with $k+1$ vertices
  2. A set Q of $\frac{1}{2}(k+1)!$ paths, which is a minimal cover,

  such that $LP'(H,S,Q)$ has a solution.

- Make a coloring of G (with $k+1$ colors).

- Check if a graph H from this list is an induced subgraph of G (preserving colors). If G does have a subgraph from the list, G is not superperfect, otherwise it is.

Notice that generating the list takes constant time (if k is a constant). Since the subgraphs have constant size, we can check if such a subgraph is an induced subgraph of G in linear time, using standard techniques for (partial) k-trees (see [1]).

Theorem 4.3 *The algorithm correctly determines if G is superperfect, and does so in linear time.*

Proof:
Assume G is not superperfect. Let C be a coloring of G. By theorem 4.2, $LP(G,S,Q)$ has a solution for some maximal clique S and some minimal cover Q. Take H the colored subgraph induced by vertices of S and of paths in Q. By lemma 4.7, $LP'(H,S,Q)$ has a solution, so the subgraph H is in the list. Conversely, suppose the colored graph G has a colored induced subgraph H from the list. Then H has a clique S with $k+1$ vertices and a minimal cover Q such that $LP'(H,S,Q)$ has a solution. Since H is an induced subgraph of G preserving colors, the clique S is also a maximal clique in G and the cover Q is also a cover for G. Since $LP(G,S,Q)$ has a solution, G can not be superperfect. □

Notice that the list only has to contain those subgraphs H, of which every vertex is either in the maximal clique S or on some path in Q (with S and Q as defined in the algorithm). For reasons of simplicity, we left this detail out of the algorithm.

5 Conclusions

In this paper we presented a linear time algorithm to test superperfection of k-trees, for some constant k. We also showed there exists a list of constant size, of forbidden colored triangulated graphs, such that G is superperfect if and only if the colored graph G does not have an induced subgraph (preserving colors) from this list. Finally, we completely characterized the 2-trees which are superperfect, by means of forbidden induced subgraphs. To test superperfection on *partial* k-trees remains an open problem.

6 Acknowledgements

We like to thank Prof. D. Seese for valuable suggestions.

References

[1] S. Arnborg, J. Lagergren and D. Seese, Easy problems for tree-decomposable graphs, *J. Algorithms*, **12**, 308-340, 1991.

[2] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees. *Disc. Appl. Math.*, **23**, 11-24, 1989.

[3] H.L. Bodlaender and T. Kloks, Better algorithms for the pathwidth and treewidth of graphs, *Proceedings of the* 18*th International colloquium on Automata, Languages and Programming*, 544-555, Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[4] C. Berge and C. Chvatal, *Topics on perfect graphs*, Annals of Discrete Mathematics **21** 1984.

[5] K.S. Booth and G.S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, *Journal of Computer and System Sciences* **13**, 335 – 379, 1976.

[6] J.E. Cohen, J. Komlós and T. Mueller, The probability of an interval graph, and why it matters, *Proc. of Symposia in Pure Math.* **34**, 97 – 115, 1979.

[7] P.C. Fishburn, An interval graph is not a comparability graph, *J. Combin. Theory* **8**, 442 – 443, 1970.

[8] D.R. Fulkerson and O.A. Gross, Incidence matrices and interval graphs, *Pacific J. Math.* **15**, 835 – 855, 1965.

[9] T. Gallai, Transitiv orientierbaren Graphen, *Acta Math. Sci. Hung.* **18**, 25 – 66, 1967.

[10] P.C. Gilmore and A.J. Hoffman, A characterization of comparability graphs and of interval graphs, *Canad. J. Math.* **16**, 539 – 548, 1964.

[11] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[12] J. Lagergren and S. Arnborg, Finding minimal forbidden minors using a finite congruence, *Proceedings of the* 18*th International colloquium on Automata, Languages and Programming*, 532-543, Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[13] L. Lovász, Normal hypergraphs and the perfect graph conjecture, *Discrete Math.*, **2**, 253 – 267, 1972.

[14] J. Matoušek and R. Thomas, Algorithms finding tree-decompositions of graphs, *Journal of Algorithms* **12**, 1-22, 1991.

[15] M. Jean, An interval graph is a comparability graph, *J. Combin. Theory* **7**, 189 – 190, 1969.

[16] A. Pnuelli, A. Lempel, and S. Even, Transitive orientation of graphs and identification of permutation graphs, *Canad. J. Math.* **23**, 160 – 175, 1971.

[17] B.A. Reed, Finding approximate separators and computing treewidth quickly, To appear in STOC'92.

[18] N. Robertson and P.D. Seymour, Graph minors — a survey. In I. Anderson, editor, *Surveys in Combinatorics* 153-171. Cambridge Univ. Press 1985.

[19] J. Spinrad, On comparability and permutation graphs, *SIAM J. Comp.* **14**, No. 3, August 1985.