On 2-Subcolourings of Chordal Graphs

Juraj Stacho

School of Computing Science, Simon Fraser University
8888 University Drive, Burnaby, B.C., Canada V5A 1S6
jstacho@cs.sfu.ca

Abstract. A 2-subcolouring of a graph is a partition of the vertices into two subsets, each inducing a P_3-free graph, i.e., a disjoint union of cliques. We give the first polynomial time algorithm to test whether a chordal graph has a 2-subcolouring. This solves (for two colours) an open problem of Broersma, Fomin, Nešetřil, and Woeginger, who gave an $O(n^5)$ time algorithm for interval graphs. Our algorithm for the larger class of chordal graphs has complexity only $O(n^3)$.

1 Introduction

A *k-subcolouring* of a graph G is a partition of the vertices of G into k subsets $V(G) = V_1 \cup V_2 \cup \ldots \cup V_k$, such that each V_i induces a disjoint union of *cliques* (complete graphs) in G, i.e., each V_i induces a P_3-free graph. A graph G is called *k-subcolourable* if there exists a k-subcolouring of G. The smallest integer k such that G is k-subcolourable is called the *subchromatic number* of G, and is denoted by $\chi_s(G)$.

The k-subcolourings and the subchromatic number were first introduced by Albertson, Jamison, Hedetniemi and Locke in [1]. Initially, the main focus was on bounds for $\chi_s(G)$. More recently, the complexity of recognizing k-subcolourable graphs has become a focus of attention. It follows from the result in [2] that for $k \geq 2$ this problem is NP-complete for general graphs. In [3] (and also in [4]) the authors show that it remains NP-complete for $k \geq 2$ even if the graph is triangle-free and of maximum degree four. On the other hand, there are several natural classes of graphs for which the problem has a polynomial time solution for any fixed k, e.g., graphs of bounded treewidth [3]. In another paper [5], the authors show that the problem is NP-complete for $k \geq 2$ when restricted to the class of comparability graphs, whereas for interval graphs there is an $O(n^{2k+1})$ time algorithm. In fact, it is easy to check that their algorithm also works for the case of *list k-subcolouring*, where each vertex of the input graph G has a list of admissible colours and the task is to determine whether or not there exists a k-subcolouring of G that obeys these lists. In this paper, we also deal with list k-subcolourings.

In [5], the authors formulated the following open problem. Determine the complexity of the k-subcolouring problem for the class of chordal graphs. This seems interesting, since the class of chordal graphs is strictly between the class of perfect graphs (for which the problem is NP-complete) and the class of interval graphs (for which the problem is polynomial time solvable), and colouring problems for chordal graphs often lead to interesting insights [6–8]. In this paper, we develop a novel technique that allows us to extract the essential properties of a 2-subcolouring, to solve this problem for $k = 2$.

In the following, we give a polynomial time algorithm for testing list 2-subcolourability of chordal graphs. In fact, our algorithm is $O(n^3)$; that also improves the complexity of the algorithm from [5] for the smaller class of interval graphs.

Instead of considering a k-subcolouring of G as a partition $V(G) = V_1 \cup V_2 \cup \ldots \cup V_k$, one can view it as a mapping $c : V(G) \to \{1, 2, \ldots, k\}$, where for every $i \in \{1, 2, \ldots, k\}$, the vertices of V_i are mapped to i. Therefore we shall employ the terminology of colourings and refer to c as a colouring of G, and refer to the elements of $\{1, 2, \ldots, k\}$ as colours. (Note that c is not necessarily a proper colouring.) For a 2-subcolouring $V(G) = V_{\mathsf{r}} \cup V_{\mathsf{b}}$, the associated colouring c is a mapping $c : V(G) \to \{\mathsf{r}, \mathsf{b}\}$, and we refer to the elements of V_{r} and V_{b} as red and blue vertices respectively.

A graph is *chordal* if it does not contain an induced cycle of length more than three [8, 9]. A *clique-tree* T of a chordal graph G is a tree with the following properties [8, 9].

(i) Each vertex u in T corresponds to a maximal clique C_u of G
(ii) For every edge $ab \in E(G)$, there exists a vertex $u \in V(T)$ such that $a, b \in C_u$
(iii) For every vertex $a \in V(G)$, the set of vertices u of $V(T)$ such that $a \in C_u$ induces a connected subgraph of T.

As usual, we denote by n and m the number of vertices and edges of an input graph G respectively. It is known [9] that recognizing a chordal graph, and constructing a clique-tree of a (connected) chordal graph, can both be performed in time $O(n + m)$.

The paper is structured as follows. Before describing our algorithm we investigate, in sections 2 and 3, the general properties of subcolourings of chordal graphs. In particular, in section 2, we introduce the key structure, called the *subcolouring digraph*, that encodes important properties of a given subcolouring of a chordal graph G (based on a fixed clique-tree of G). In section 3, we describe the necessary conditions for

a 2-subcolouring c implied by the structure of its subcolouring digraph. Finally, in section 4, we describe our algorithm, which uses dynamic programming on the subcolouring digraph, and we discuss the complexity and efficient implementation of our algorithm.

2 The Subcolouring Digraph

Observe first that G is k-subcolourable if and only if each component of G is k-subcolourable. Throughout the paper, unless otherwise indicated, we shall always deal with a connected chordal graph G, a fixed clique-tree T of G, and with c, a colouring of the vertices of G (not necessarily a subcolouring or a proper colouring).

Therefore, let G be a connected chordal graph, and let T be a fixed clique-tree of G. Let C_u for $u \in V(T)$ denote the maximal clique of G associated with u, and let $\mathcal{C}(X)$ denote the union of cliques associated with the vertices of a set $X \subseteq V(T)$, i.e., $\mathcal{C}(X) = \bigcup_{u \in X} C_u$. In this section, we shall not consider T to be rooted. We shall use parentheses $(,)$ to denote the edges of T, to distinguish them from the edges of G. The removal of an edge (u, v) splits T into two subtrees; we shall denote by $T_{u,v}$ the subtree containing the vertex v, and by $T_{v,u}$ the subtree containing the vertex u. We denote $G_{u,v} = \mathcal{C}(T_{u,v})$ and $G_{v,u} = \mathcal{C}(T_{v,u})$.

Observe that in any k-subcolouring c of G, a vertex a in a clique C of G can have neighbours of the same colour as a in at most one connected component of $G \setminus C$. Based on this, we construct a multidigraph $\mathcal{D}_c(G)$ with coloured edges to capture the properties of the colouring c. We shall refer to $\mathcal{D}_c(G)$ as a *subcolouring digraph* for c. To avoid ambiguity, the edges of $\mathcal{D}_c(G)$ shall be referred to as *arcs* and denoted using angle brackets $\langle, \rangle$ to distinguish them from the edges of T and the edges of G. In particular, $\langle u, v \rangle_i$ shall denote an arc from u to v coloured i, and we shall write $\langle u, v \rangle$ for an arc from u to v (of some colour). The digraph $\mathcal{D}_c(G)$ is constructed as follows (cf. Figure 1). The vertices of $\mathcal{D}_c(G)$ are the vertices of T, and for vertices u, v that are adjacent in T, there is an arc $\langle u, v \rangle_i$ in $\mathcal{D}_c(G)$, if there exist vertices $a \in C_u$ and $b \in G_{u,v} \setminus C_u$ such that $ab \in E(G)$ and both a and b have the same colour i in c. Note that we have arcs in $\mathcal{D}_c(G)$ only between vertices that are adjacent in T.

Formally, we define $\mathcal{D}_c(G)$ as follows.

(i) $V(\mathcal{D}_c(G)) = V(T)$

(ii) $E(\mathcal{D}_c(G)) = \left\{ \langle u, v \rangle_i \;\middle|\; \begin{array}{ll} \exists\, a \in C_u & (u, v) \in E(T) \\ \exists\, b \in G_{u,v} \setminus C_u & ab \in E(G) \\ & c(a) = c(b) = i \end{array} \right\}$

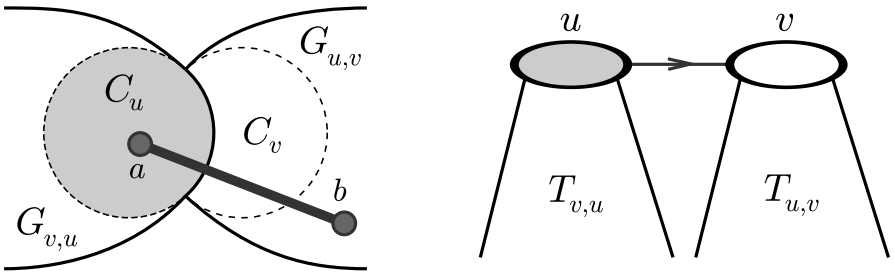

Fig. 1. Illustrating the case when there is an arc $\langle u, v\rangle$ in $\mathcal{D}_c(G)$

We have the following observations about $\mathcal{D}_c(G)$.

Proposition 1. *Let u, v, w be vertices of $\mathcal{D}_c(G)$ and let i be a colour from $\{1, 2, \ldots, k\}$. If c is a k-subcolouring then $\mathcal{D}_c(G)$*

(i) cannot contain both the arc $\langle u, v\rangle_i$ and the arc $\langle v, u\rangle_i$,
(ii) cannot contain both the arc $\langle u, v\rangle_i$ and the arc $\langle u, w\rangle_i$,
(iii) cannot contain all of the arcs $\langle u, v\rangle_1, \langle u, v\rangle_2, \ldots, \langle u, v\rangle_k$.

Proof. Suppose that (i) is false. Let a, b and a', b' be the vertices of G that caused the arcs $\langle u, v\rangle_i$ and $\langle v, u\rangle_i$ respectively to appear in $\mathcal{D}_c(G)$. It is not difficult to see that $a, a' \in C_u \cap C_v$ and $bb' \notin E(G)$. Hence the graph induced on a, b, a', b' must contain an induced P_3 coloured i, a contradiction. One can easily repeat this same argument for pairs a, b and a', b' that falsify (ii).

Finally, let a_1, b_1, a_2, b_2, $\ldots$, a_k, b_k be the pairs of vertices that falsify (iii). Since C_u and C_v are two different maximal cliques of G, there exists a vertex $d \in C_u \setminus C_v$. Again, it is not difficult to see that $a_i \in C_u \cap C_v$ for all i, and d is not a neighbour of any b_i. Now clearly, any colour j assigned to d creates an induced P_3 coloured j on vertices d, a_j, b_j, yielding a contradiction. □

3 2-Subcolourings

From now on we focus on the case $k = 2$, i.e., 2-subcolourings. In what follows, we shall assume that G is a connected 2-subcolourable chordal graph, T a fixed clique-tree of G, and c is a 2-subcolouring of G. As remarked earlier, for a 2-subcolouring c of G, we shall refer to the vertices of G as red and blue vertices and use letters r and b respectively to denote the two colours.

We shall call an edge (u, v) in T a *strong* edge of T if $\mathcal{D}_c(G)$ contains both $\langle u, v\rangle_{\mathsf{r}}$ and $\langle v, u\rangle_{\mathsf{b}}$, or both $\langle u, v\rangle_{\mathsf{b}}$ and $\langle v, u\rangle_{\mathsf{r}}$. We shall call an edge (u, v) in T a *weak* edge of T if there is at most one arc between u and v in $\mathcal{D}_c(G)$. It follows from Proposition 1 that every edge in T must be either strong or weak.

Let u, v be adjacent vertices in T. Let $I_{u,v} = C_u \cap C_v$, and let $N_{u,v}$ be the set of all vertices of $G_{u,v} \setminus C_u$ which are neighbours of $C_u \cap C_v$. Furthermore, let $L_{u,v} = G_{v,u} \setminus C_v$ and $R_{u,v} = G_{u,v} \setminus (I_{u,v} \cup N_{u,v})$. (Note that $C_v \subseteq I_{u,v} \cup N_{u,v}$.) We have the following observation.

Proposition 2. *Let u, v be adjacent vertices in T.*

(i) If $\langle u, v\rangle_{\mathsf{r}} \in E(\mathcal{D}_c(G))$, or $\langle u, v\rangle_{\mathsf{b}} \in E(\mathcal{D}_c(G))$, then the vertices of $C_u \setminus C_v$ are all blue, or all red, respectively.

(ii) If $\langle u, v\rangle \notin E(\mathcal{D}_c(G))$ then the vertices of $I_{u,v}$ and $N_{u,v}$ are all red and all blue respectively, or all blue and all red respectively.

(iii) G has no induced P_3 having both a vertex of $L_{u,v}$ and a vertex of $R_{u,v}$.

Proof. For (i) suppose that $\langle u, v\rangle_{\mathsf{r}} \in E(\mathcal{D}_c(G))$ (the other case is clearly symmetric), and let a, b be the red vertices that caused this arc. It is easy to see, that $a \in C_u \cap C_v$ and b is not adjacent to any vertex in $C_u \setminus C_v$. Hence, no vertex d of $C_u \setminus C_v$ can be red, since otherwise d, a, b is an induced red P_3.

For (ii) let $a \in I_{u,v}$ and $b \in N_{u,v}$ be adjacent. Then a and b must have different colours, since otherwise we would have an arc $\langle u, v\rangle \in E(\mathcal{D}_c(G))$. Since C_u and C_v are different maximal cliques, there exists $d \in C_v \setminus C_u$. Now the claim follows, because d is adjacent to all vertices of $I_{u,v}$, and hence any $a' \in I_{u,v}$ must have different colour from d.

Finally, for (iii) let b, a, d be an induced P_3 in G with edges ba and ad, that contains both a vertex of $L_{u,v}$ and a vertex of $R_{u,v}$. Now since $L_{u,v}$ and $R_{u,v}$ are completely non-adjacent, it follows that $a \notin L_{u,v} \cup R_{u,v}$. Hence we can assume that $b \in L_{u,v}$ and $d \in R_{u,v}$. Now if $a \in I_{u,v}$ then we must have $d \in N_{u,v}$ but $N_{u,v} \cap R_{u,v} = \emptyset$. Hence $a \in N_{u,v}$ and $b \in I_{u,v}$ but $L_{u,v} \cap I_{u,v} = \emptyset$. Therefore no such vertices b, a, d exist in G. □

Note that it follows from the above observation that for an edge (u, v) in T, the 2-subcolourings induced by the fixed c on $L_{u,v}$ and $R_{u,v}$, are independent of each other in the sense that they only depend on the colours assigned to $I_{u,v} \cup N_{u,v}$. Furthermore, if (u, v) is a weak edge such that $\langle u, v\rangle \notin E(\mathcal{D}_c(G))$, then the 2-subcolouring of $I_{u,v} \cup N_{u,v}$ is unique (up to exchanging the colours red and blue). That allows one to consider independently the subgraphs of T that no longer contain any weak edges.

For strong edges in T we have the following observations.

Observation 3. *Every vertex of T has at most two incident strong edges, i.e., the connected components formed by the strong edges of T are paths.*

Proof. It is not difficult to see that if a vertex u in T has three adjacent strong edges, then for at least two of them, say (u, v) and (u, w), we have arcs $\langle u, v\rangle$ and $\langle u, w\rangle$ of the same colour in $\mathcal{D}_c(G)$. By Proposition 1(*ii*) this is not possible. □

Proposition 4. *Let u be a vertex in T with distinct neighbours v, w, z.*

(i) If (v, u) is a strong edge and $\langle w, u\rangle \notin E(\mathcal{D}_c(G))$, then (z, u) is not a strong edge.
(ii) If (v, u) is a strong edge, $\langle w, u\rangle \notin E(\mathcal{D}_c(G))$, and $\langle z, u\rangle \notin E(\mathcal{D}_c(G))$, then $I_{u,w} = I_{u,z}$.
(iii) If $\langle v, u\rangle \notin E(\mathcal{D}_c(G))$, $\langle w, u\rangle \notin E(\mathcal{D}_c(G))$, and $\langle z, u\rangle \notin E(\mathcal{D}_c(G))$, then $I_{u,v} = I_{u,w}$ or $I_{u,w} = I_{u,z}$ or $I_{u,z} = I_{u,v}$.

Proof. For (i) suppose that the edges (v, u) and (z, u) are strong and that $\langle w, u\rangle \notin E(\mathcal{D}_c(G))$. Since G is connected, there exists $a \in I_{u,w}$. Without loss of generality we may assume that $\langle u, z\rangle_{\mathsf{r}} \in E(\mathcal{D}_c(G))$. Hence by Proposition 2$(i)$ we obtain that $C_z \setminus C_v$ is all red, $C_v \setminus C_z$ is all blue and $C_u \subseteq C_v \cup C_z$. Also since C_u, C_z and C_v are different maximal cliques we have $d \in C_z \setminus C_u$ and $b \in C_v \setminus C_u$. Now clearly, $a \in C_v \cup C_z$ hence if a is red then $a \in C_z$ and hence $\langle w, u\rangle_{\mathsf{r}} \in E(\mathcal{D}_c(G))$, and if a is blue then $a \in C_v$ and hence $\langle w, u\rangle_{\mathsf{b}} \in E(\mathcal{D}_c(G))$, a contradiction.

For (ii) suppose that (v, u) is strong, $\langle w, u\rangle \notin E(\mathcal{D}_c(G))$, and $\langle z, u\rangle \notin E(\mathcal{D}_c(G))$ but $I_{u,w} \neq I_{u,z}$. Without loss of generality we may assume that $I_{u,w} \not\subseteq I_{u,z}$ and that $\langle u, v\rangle_{\mathsf{r}}, \langle v, u\rangle_{\mathsf{b}} \in E(\mathcal{D}_c(G))$. Hence there must exist a vertex $a \in I_{u,w} \setminus I_{u,z}$, a vertex $b \in I_{u,z}$ (since G is connected), and a vertex $d \in C_v \setminus C_u$ (since the cliques are maximal). Clearly, $c(a) \neq c(b)$ otherwise we have $\langle z, u\rangle \in E(\mathcal{D}_c(G))$. Now since $\langle v, u\rangle_{\mathsf{b}} \in E(\mathcal{D}_c(G))$ it follows that the vertex d is red. Similarly, since $\langle u, v\rangle_{\mathsf{r}} \in E(\mathcal{D}_c(G))$ we have that $C_u \setminus C_v$ is all blue, hence if a is red then $a \in I_{u,v}$ and hence $\langle w, u\rangle_{\mathsf{r}} \in E(\mathcal{D}_c(G))$, and if b is red then $b \in I_{u,v}$ and hence $\langle z, u\rangle_{\mathsf{r}} \in E(\mathcal{D}_c(G))$, a contradiction.

Finally, for (iii) suppose that none of $\langle v, u\rangle$,$\langle w, u\rangle$, and $\langle z, u\rangle$ is in $E(\mathcal{D}_c(G))$, but the three sets $I_{u,v}$, $I_{u,w}$ and $I_{u,z}$ are pairwise different. Without loss of generality we may assume that $I_{u,v} \not\subseteq I_{u,w} \not\subseteq I_{u,z}$ and either $I_{u,v} \not\subseteq I_{u,z}$ or $I_{u,v} \not\supseteq I_{u,z}$. If $I_{u,v} \not\subseteq I_{u,z}$ suppose first that $J \neq \emptyset$ where $J = I_{u,v} \setminus (I_{u,w} \cup I_{u,z})$. It follows that we must have a vertex $a \in J$, a

vertex $b \in I_{u,w} \setminus I_{u,z}$ and a vertex $c \in I_{u,z}$. Now at least two of the vertices a, b, c must have the same colour and that gives us one of the edges $\langle v, u\rangle$, $\langle w, u\rangle$, $\langle z, u\rangle$ in $E(\mathcal{D}_c(G))$, a contradiction. If $J = \emptyset$ we similarly obtain a contradiction for vertices $a \in (I_{u,w} \cap I_{u,v}) \setminus I_{u,z}$, $b \in (I_{u,z} \cap I_{u,v}) \setminus I_{u,w}$ and $c \in C_u \setminus C_v$. Now if $I_{u,v} \not\supseteq I_{u,z}$ it follows that we have a vertex $a \in I_{u,v} \setminus I_{u,w}$, a vertex $b \in I_{u,w} \setminus I_{u,z}$ and $c \in I_{u,z} \setminus I_{u,v}$ and again a contradiction follows. □

Let $P_{u,v}$ denote the (unique) path from u to v in T. We shall call the path $P_{u,v}$ *strong* if it is formed only by strong edges of T. Note that we also allow paths of zero length (i.e., paths $P_{u,v}$ with $u = v$); all such paths are trivially strong. A strong path is *maximal* if it is not properly contained in another strong path. A vertex z in T adjacent to a vertex u is a *special neighbour* of u if $\langle z, u\rangle \notin E(\mathcal{D}_c(G))$. The following claim is a direct consequence of Proposition 4.

Lemma 5. *For any strong path $P_{u,v}$ in T (possibly with $u = v$) there exist sets $A_{u,v}$ and $A'_{u,v}$ (both possibly empty) such that for any special neighbour s of some $t \in P_{u,v}$ we have $I_{s,t} = A_{u,v}$ or $I_{s,t} = A'_{u,v}$.*

Proof. If $u \neq v$ then by Proposition 4(i) only u and v can have special neighbours. Hence, if u has a special neighbour z, we let $A_{u,v} = I_{z,u}$ and $A_{u,v} = \emptyset$ otherwise. If v has a special neighbour w, we let $A'_{u,v} = I_{w,v}$ and $A'_{u,v} = \emptyset$ otherwise. Now the claim follows from Proposition 4(ii).

If $u = v$ and u has two special neighbours z and w with $I_{z,u} \neq I_{w,u}$, we define $A_{u,v} = I_{z,u}$, $A'_{u,v} = I_{w,u}$. Otherwise, we let $A_{u,v} = I_{z,u}$ and $A'_{u,v} = \emptyset$ if u has a special neighbour z but does not satisfy the previous condition. Finally, we let $A_{u,v} = A'_{u,v} = \emptyset$ if u has no special neighbours. Now the claim follows from Proposition 4(iii). □

Let $B_{u,v}$ and $B'_{u,v}$ denote the sets of neighbours of $A_{u,v}$ and $A'_{u,v}$ in $\mathcal{C}(P_{u,v})$ respectively. We now give a complete characterisation of the structure of the colouring c on the vertices of $\mathcal{C}(P_{u,v})$.

Theorem 6. *For any strong path $P_{u,v}$ in T (possibly with $u = v$) we have*

- *(i)* $\mathcal{C}(P_{u,v}) = C_u \cup C_v$,
- *(ii) the vertices of $C_u \setminus C_v$ and $C_v \setminus C_u$ are all red and all blue respectively, or all blue and all red respectively,*
- *(iii) for every weak edge (s, t) incident to $P_{u,v}$ the vertices of $I_{s,t}$ are all red or all blue,*
- *(iv) the vertices of $A_{u,v} \cup B'_{u,v}$ and $A'_{u,v} \cup B_{u,v}$ are all red and all blue respectively, or all blue and all red respectively, and*

(v) if in addition $P_{u,v}$ is maximal, then any colouring c' of $\mathcal{C}(P_{u,v})$ satisfying $(ii)-(iv)$ is a 2-subcolouring of $\mathcal{C}(P_{u,v})$ and can be extended to a 2-subcolouring of G.

Proof. We prove (i) and (ii) by induction on the length of the path $P_{u,v}$. If $u = v$ then there is nothing to prove. Hence, let w be the neighbour of v on $P_{u,v}$ and assume that $\mathcal{C}(P_{u,w}) = C_u \cup C_w$ and that the vertices of $C_u \setminus C_w$ and $C_w \setminus C_u$ are all red and all blue respectively. Since (w, v) is a strong edge, by Proposition $2(i)$ we have that $C_v \setminus C_w$ is all blue and $C_w \setminus C_v$ is red. From that we deduce $C_w \setminus (C_u \cup C_v) = \emptyset$ which implies $C_w \subseteq C_u \cup C_v$ and the claim follows.

Now claims (iii) and (iv) follow directly from Proposition $2(ii)$ and Lemma 5 since for any special neighbour s of $t \in P_{u,v}$ we have $I_{s,t} = A_{u,v}$ and $N_{s,t} \cap \mathcal{C}(P_{u,v}) = B_{u,v}$ or $I_{s,t} = A'_{u,v}$ and $N_{s,t} \cap \mathcal{C}(P_{u,v}) = B'_{u,v}$.

Finally, let c' be any colouring of $\mathcal{C}(P_{u,v})$ satisfying $(ii) - (iv)$. Let c'' be a colouring of G constructed from the colouring c as follows. First we exchange the colours red and blue on the vertices of $G_{t,s}$ for each neighbour $s \notin P_{u,v}$ of $t \in P_{u,v}$ such that the colours of $I_{s,t}$ match the colouring c'. (Note that since $P_{u,v}$ is maximal, the edge (s, t) is weak.) Then we replace the colours of $\mathcal{C}(P_{u,v})$ by c'. Clearly, c'' extends c'. We show that c'' is a 2-subcolouring of G. Suppose otherwise and let b, a, d be an induced P_3 in G with edges ba and ad such that $c''(b) = c''(a) = c''(d)$. If $b, a, d \in \mathcal{C}(P_{u,v})$ then by (i) and (ii) it follows that the vertices b, a, d are all in C_u or all in C_v, but that is not possible since $bd \notin E(G)$. On the other hand, if $a \notin \mathcal{C}(P_{u,v})$ then it follows from the construction of c'' that $c(b) = c(a) = c(d)$ which is not possible since c is a 2-subcolouring. Hence for some neighbour $s \notin P_{u,v}$ of $t \in P_{u,v}$ we have that $a \in I_{t,s}$ and $b \in N_{t,s}$ and $d \in N_{s,t} \cap \mathcal{C}(P_{u,v})$. Now if $\langle t, s\rangle \notin E(\mathcal{D}_c(G))$ then by Proposition $2(ii)$ we have that $c(a) \neq c(b)$ hence $c''(a) \neq c''(b)$ because $a, b \in G_{t,s}$. On the other hand, if $\langle t, s\rangle \in E(\mathcal{D}_c(G))$, then s must be a special neighbour of t but then by (iv) we have that $a \in A_{u,v}$ and $d \in B_{u,v}$ or $a \in A'_{u,v}$ and $d \in B'_{u,v}$ and hence $c''(a) \neq c''(d)$, a contradiction. □

4 The algorithm

Now we are ready to describe the algorithm for deciding (list) 2-subcolourability for chordal graphs. We assume that we are given a chordal graph G and a fixed clique-tree T of G, and we want to decide whether or not G is 2-subcolourable. Later, we describe how to obtain a list version of the algorithm.

This time, we consider T rooted at an arbitrary fixed vertex r. Therefore, we write $p[v]$ to denote the parent of a vertex v in T. For a vertex v in T, we denote by T_v the subtree of T rooted at v.

We shall say that T_v is $(-)$ colourable if there exists a 2-subcolouring c_v of $\mathcal{C}(T_v)$ such that the vertices of $I_{p[v],v}$ are all red or all blue. Similarly, T_v is $(+)$ colourable if there exists a 2-subcolouring c_v of $\mathcal{C}(T_v)$ such that the vertices of $I_{p[v],v}$ and $N_{p[v],v}$ are all red and all blue respectively, or all blue and all red respectively. In the special case of the root r, when $p[v]$ does not exist, we shall say that T_r is $(-)$ colourable if there exists a 2-subcolouring c_r of $\mathcal{C}(T_r) = G$.

Note that by Lemma 5, for every strong path, we only need to consider up to two special neighbours z and w. In case that the path has only one (or none), we use nil as the value of z or w. Therefore, we always view a path $P_{u,v}$ having two special neighbours z and w of u and v respectively, but allow one (or both) of z and w to be nil. We shall say that the path $P_{u,v}$ is (z,w)-colourable if there exists a 2-subcolouring $c_{u,v}$ of $\mathcal{C}(P_{u,v})$ such that for every incident edge (s,t) of $P_{u,v}$ in T, the vertices of $I_{s,t}$ are all red or all blue, and such that if z is not nil (w is not nil) then the vertices of $N_{z,u}$ ($N_{w,v}$ respectively) in $\mathcal{C}(P_{u,v})$ are all red or all blue.

The algorithm works as follows. It processes the vertices of T in a bottom-up order and identifies which edges of T could be weak, for some 2-subcolouring of G. This is done by testing and recording for every vertex v in T, whether or not the subtree T_v is $(+)$ colourable, and whether or not the subtree T_v is $(-)$ colourable. (Note that T_v must be either $(+)$ colourable or $(-)$ colourable if $(v, p[v])$ is a weak edge in T for some 2-subcolouring of G.)

For a vertex x of T, the test for colourability of T_x is done as follows. First, if we are testing $(-)$ colourability, we precolour the vertices of $I_{p[x],x}$ red or blue, otherwise we precolour the vertices of $I_{p[x],x}$ and $N_{p[x],x}$ red and blue respectively, or blue and red respectively. Then we choose a strong path $P_{u,v}$ in T_x that passes through x and we choose special neighbours z and w of u and v respectively. (See the above remark about special neighbours.) Then we test for colourability of $P_{u,v}$ with respect to the chosen special neighbours by applying Theorem 6. Finally, we recursively test, for every special neighbour y of $P_{u,v}$, whether or not the corresponding tree T_y is $(-)$ colourable or $(+)$ colourable, and for all other neighbours of $P_{u,v}$, whether or not their corresponding trees are $(+)$ colourable. We declare T_x $(-)$ colourable (or $(+)$ colourable, depending on the particular case) if and only if the above tests succeed for some choice of u, v and some choice of special neighbours of u and v. Note that

since we process the vertices in a bottom-up order, each recursive call amounts to a constant time table look-up.

If the algorithm succeeds to declare T_r $(-)$ colourable then the graph G is 2-subcolourable, otherwise G is not 2-subcolourable. The correctness can be shown to follow from Proposition 2 and Theorem 6. A more precise description of an efficient implementation of the algorithm can be found on pages 11-12. Below, we discuss some details of this implementation.

Note that in the procedure for testing colourability of a strong path $P_{u,v}$ (see Algorithm 2 on page 11), instead of independently precolouring the sets $I_{s,t}$ either red or blue, for each incident edge of $P_{u,v}$ (as follows from Theorem 6), we construct a collection of sets $\mathcal{L}$ containing the unions of the sets $I_{s,t}$ that intersect. Note that if sets $I_{s,t}$ and $I_{s',t'}$ intersect, their union must also be all red or all blue. After constructing $\mathcal{L}$ we can independently decide the colours of the sets in $\mathcal{L}$, since they no longer intersect. To find $\mathcal{L}$ we use an efficient variant of the Union-Find algorithm which has time and space complexity $O(n)$.

It is not difficult to see that this algorithm can be easily extended to solve the list 2-subcolouring problem. Recall that this is the problem where each vertex has a list of admissible colours and the task is to decide whether G has a 2-subcolouring that obeys these lists. Whenever in the algorithm a vertex is being precoloured by some colour, this colour is checked against the list of that vertex and if it is not in the list, we exit the current procedure with a negative answer. (Note that this only happens in the procedure for testing strong paths, see Algorithm 2 on page 11.)

The following theorem summarizes the complexity of the above algorithm and is followed by its formal description.

Theorem 7. *There exists an $O(n^3)$ time algorithm deciding for a given chordal graph G, whether G admits a (list) 2-subcolouring; the algorithm also constructs a 2-subcolouring of G if one exists.*

Proof. As noted before, one can determine the maximal cliques of G and construct a clique-tree T of G in time $O(n+m)$. It follows from the remark above that for any pair of vertices u, v and choice of special neighbours z, w of u, v respectively, one can determine (z, w)-colourability of the path $P_{u,v}$ in time $O(n^2)$. In the first part of the algorithm, this test is performed for every pair of vertices (including the choice of their special neighbours). This step can be implemented more efficiently by reusing the results for the subpaths, i.e., starting from some vertex v and computing all paths from v, altogether in time $O(n^2)$. Therefore the total running time for the first part of the algorithm is $O(n^3)$. In the second part, note that during

the course of the algorithm (in the procedures for testing the colourability of a subtree T_x, see Algorithms 3,4 on page 12), we consider every path (including the choice of special neighbours) in T only once. Each path is processed in time $O(n)$, so in total we have $O(n^3)$ time. Finally, a 2-subcolouring can be easily found by keeping track of which paths were used to colour the subtrees of T and backtracking from the root r. □

Input: A chordal graph G and a clique tree T of G rooted at r
Output: Decide whether G is 2-subcolourable

for every two vertices u, v in T **do**
    for every neighbour z and w (including nil) of u and v respectively **do**
        test and record whether $P_{u,v}$ is (z, w)-colourable
initialize $S \leftarrow \emptyset$ (S is the set of processed vertices)
while $S \neq V(T)$ **do**
    pick a vertex $v \notin S$ whose all children are in S
    test and record whether T_v is $(-)$ colourable
    test and record whether T_v is $(+)$ colourable (if $v \neq r$)
    $S \leftarrow S \cup \{v\}$
if T_r is $(-)$ colourable **then**
  return *"G is 2-subcolourable"*
else return *"G is not 2-subcolourable"*

Algorithm 1: The test for 2-subcolourability of G

Input: Vertices u, v of T, vertices z, w neighbours of u, v respectively or nil
Output: Decide whether $P_{u,v}$ is (z, w)-colourable

compute $\mathcal{C}(P_{u,v})$
if $\mathcal{C}(P_{u,v}) \neq C_u \cup C_v$ **then return** *"$P_{u,v}$ is not (z, w)-colourable"*
precolour the vertices of $C_u \setminus C_v$ and $C_v \setminus C_u$ by red and blue respectively (or blue and red respectively)
if $z \neq$ nil **then** precolour the vertices of $N_{z,u}$ red (or blue)
if $w \neq$ nil **then** precolour the vertices of $N_{w,v}$ blue (or red)
initialize the set $\mathcal{L} \leftarrow \emptyset$
for each edge (s, t) incident to $P_{u,v}$ **do**
    compute the set $\mathcal{L}_s$ consisting of those sets from $\mathcal{L}$ which intersect $I_{s,t}$
    $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{L}_s \cup \left\{ I_{s,t} \cup \left(\bigcup_{L \in \mathcal{L}_s} L \right) \right\}$
if some $L \in \mathcal{L}$ contains both a precoloured red and a precoloured blue vertex **then return** *"$P_{u,v}$ is not (z, w)-colourable"*
else return *"$P_{u,v}$ is (z, w)-colourable"*

Algorithm 2: The test whether $P_{u,v}$ is (z, w)-colourable.

```
   Input: A vertex x in T
   Output: Decide whether T_x is (−) colourable
1  for each u, v ∈ T_x such that x ∈ P_{u,v} do
2      for every child z and w (including nil) of u and v respectively do
3          if P_{u,v} is (z, w)-colourable
             and for each child s ≠ w, z of P_{u,v} the tree T_s is (+) colourable
               and either z = nil (resp. w = nil) or T_z (resp. T_w) is (+) or (−)
                 colourable then return "T_x is (−) colourable"
4  return "T_x is not (−) colourable"
```

Algorithm 3: The test whether T_x is $(-)$ colourable.

```
   Input: A vertex x in T
   Output: Decide whether T_x is (+) colourable
1  for each u ∈ T_x do
2      for every child z (including nil) of u do
3          if P_{u,x} is (z, p[x])-colourable
             and for each child s ≠ z of P_{u,x} the tree T_s is (+) colourable
               and either z = nil or T_z is (+) or (−) colourable
                 then return "T_x is (+) colourable"
4  return "T_x is not (+) colourable"
```

Algorithm 4: The test whether T_x is $(+)$ colourable.

Acknowledgements

The author would like to thank his advisor Pavol Hell for directing this research and for his help with the preparation of this paper.

Note added in proof

The algorithm presented in this paper answers an open question from [5] for the case $k = 2$ while it also extends a result from [5] and improves the complexity for the larger class of chordal graphs. Recently, we were able to show that for all other values of $k \geq 3$, the problem of k-subcolouring of chordal graphs is NP-complete, thus completely answering the open question of Broersma et al. [5].

References

1. M. O. Albertson, R. E. Jamison, S. T. Hedetniemi, S. C. Locke: *The subchromatic number of a graph*, Discrete Mathematics 74 (1989), 33–49.

2. D. Achlioptas: *The complexity of G-free colorability*, Discrete Mathematics 165/166 (1997), 21–30.
3. J. Fiala, K. Jansen, V. B. Le, and E. Seidel: *Graph subcoloring: Complexity and algorithms*, In: Graph-Theoretic Concepts in Computer Science (WG 2001), Springer, Berlin, 2001, 154–165.
4. J. Gimbel, C. Hartman: *Subcolorings and the subchromatic number of a graph*, Discrete Mathematics 272 (2003), 139–154.
5. H. Broersma, F. V. Fomin, J. Nešetřil, G. J. Woeginger: *More about subcolorings*, Computing 69 (2002), 187–203.
6. P. Hell, S. Klein, L.T. Nogueira, F. Protti: *Partitioning chordal graphs into independent sets and cliques*, Discrete Applied Mathematics 141 (2004), 185–194.
7. T. Feder, P. Hell, S. Klein, L.T. Nogueira, F. Protti: *List matrix partitions of chordal graphs*, Theoretical Computer Science 349 (2005), 52–66.
8. M. C. Golumbic: **Algorithmic Graph Theory and Perfect Graphs**, Academic Press, New York, 1980.
9. J. P. Spinrad: **Efficient Graph Representations**, American Mathematical Society, 2003.