Computing role assignments of chordal graphs[*]

Pim van 't Hof[1], Daniël Paulusma[1] and Johan M.M. van Rooij[2]

[1]Department of Computer Science, University of Durham,
Science Laboratories, South Road, Durham DH1 3LE, England.
{pim.vanthof,daniel.paulusma}@durham.ac.uk
[2]Department of Information and Computing Sciences, Universiteit Utrecht,
PO Box 80.089, 3508TB Utrecht, The Netherlands.
jmmrooij@cs.uu.nl

Abstract. In social network theory, a simple graph G is called k-role assignable if there is a surjective mapping that assigns a number from $\{1, \ldots, k\}$ called a role to each vertex of G such that any two vertices with the same role have the same sets of roles assigned to their neighbors. The decision problem whether such a mapping exists is called the k-Role Assignment problem. This problem is known to be NP-complete for any fixed $k \geq 2$. In this paper we classify the computational complexity of the k-Role Assignment problem for the class of chordal graphs. We show that for this class the problem becomes polynomially solvable for $k = 2$, but remains NP-complete for any $k \geq 3$. This generalizes results of Sheng and answers his open problem.

1 Introduction

Given two graphs, say G on vertices $u_1, \ldots, u_n$ and R on vertices $1, \ldots, k$ called *roles*, an *R-role assignment* of G is a vertex mapping $r : V_G \to V_R$ such that the neighborhood relation is maintained, i.e., all neighbors of a vertex u's role $r(u)$ in R appear as roles of vertices in the neighborhood of u in G. Such a condition can be formally expressed as

$$\text{for all } u \in V_G : r(N_G(u)) = N_R(r(u)),$$

where $N_G(u)$ denotes the set of neighbors of u in the graph G. An R-role assignment r of G is called a *k-role assignment* of G if $|r(V_G)| = |V_R| = k$. Here, we use the shorthand notation $r(S) = \{r(u) \mid u \in S\}$ for $S \subseteq V_G$. An equivalent definition states that r is a k-role assignment of G if r maps each vertex of G into a positive integer such that $|r(V_G)| = k$ and $r(N_G(u)) = r(N_G(u'))$ for any two vertices u and u' with $r(u) = r(u')$.

Role assignments are introduced by Everett and Borgatti [8], who call them role colorings. They originate in the theory of social behavior. The *role graph* R models roles and their relationships, and for a given society (e.g., a hospital with doctors, nurses and patients) we can ask if its individuals can be assigned

[*] This work has been supported by EPSRC (EP/D053633/1).

roles such that relationships are preserved: each person playing a particular role has exactly the roles prescribed by the model among its neighbors. This way one investigates whether large networks of individuals can be compressed into smaller ones that still give some description of the large network. Because persons of the same social role may be related to each other, the smaller network can contain loops. In other words, given a *simple* instance graph G of n vertices does there exist a *possibly nonsimple* role graph R of $k < n$ vertices in such a way that G has an R-role assignment? From the computational complexity point of view it is interesting to know whether the existence of such assignment can be decided quickly (in polynomial time). This leads to the following two decision problems.

R-Role Assignment
Input: a simple graph G.
Question: does G have an R-role assignment?

k-Role Assignment
Input: a simple graph G.
Question: does G have an k-role assignment?

Known results and related work. A *graph homomorphism* from a graph G to a graph R is a vertex mapping $r : V_G \to V_R$ satisfying the property that $r(u)r(v)$ belongs to E_R whenever the edge uv belongs to E_G. If for every $u \in V_G$ the restriction of r to the neighborhood of u, i.e. the mapping $r_u : N_G(u) \to N_R(f(u))$, is bijective, we say that r is *locally bijective* [1, 16]. If for every $u \in V_G$ the mapping r_u is injective, we say that r is *locally injective* [9, 10]. If for every $u \in V_G$ the mapping r_u is surjective, r is an R-role assignment of G. In this context, r is also called a *locally surjective* homomorphism from G to R.

Locally bijective homomorphisms have applications in distributed computing [2, 3, 5] and in constructing highly transitive regular graphs [4]. Locally injective homomorphisms, also called partial graph coverings, have applications in models of telecommunication [10] and frequency assignment [11]. Besides social network theory [8, 17, 19], locally surjective homomorphisms also have applications in distributed computing [6].

The main computational question is whether for every graph R the problem of deciding if an input graph G has a homomorphism of given local constraint to the fixed graph R can be classified as either NP-complete or polynomially solvable. For the locally bijective and injective homomorphisms there are many partial results, see e.g. [10, 16] for both NP-complete and polynomially solvable cases, but even conjecturing a classification for these two locally constrained homomorphisms is problematic. This is not the case for the locally surjective constraint and its corresponding decision problem R-Role Assignment.

First of all, Roberts and Sheng [19] show that the k-Role Assignment problem is already NP-complete for $k = 2$. The authors of [12] show that the k-Role Assignment problem is also NP-complete for any fixed $k \geq 3$ and classify the computational complexity of the R-Role Assignment problem. Let R be a fixed role graph without multiple edges but possibly with self-loops. Then the R-Role Assignment problem is solvable in polynomial time if and only if one of the following three cases holds: either R has no edge, or one of its components consists of a single vertex incident with a loop, or R is simple and bipartite and has at least one component isomorphic to an edge. In all other cases the R-Role Assignment problem is NP-complete, even for the class of bipartite graphs [12]. If the instance graphs are trees, then the R-Role Assignment problem becomes polynomially solvable for any fixed role graph R [13].

A graph is *chordal* if it does not contain an induced cycle of length at least four. Chordal graphs are also called *triangulated* graphs. This class contains various subclasses such as trees, split graphs and indifference graphs (graphs whose vertices can be assigned some function value such that two vertices are adjacent if and only if their function values are sufficiently close). Due to their nice properties, chordal graphs form an intensively studied graph class both within structural graph theory and within algorithmic graph theory. Sheng [20] presents an elegant greedy algorithm that solves the 2-Role Assignment problem in polynomial time for chordal graphs with at most one vertex of degree one. He also characterizes all indifference graphs that have a 2-role assignment.

Our results. We provide a polynomial time algorithm for the 2-Role Assignment problem on chordal graphs. This settles an open problem of Sheng [20]. Contrary to the greedy algorithm of [20], which uses a perfect elimination scheme of a chordal graph with at most one pendant vertex, our algorithm works for an arbitrary chordal graph G by using a dynamic programming procedure on a clique tree decomposition of G. Our second result states that, for any fixed $k \geq 3$, the k-Role Assignment problem remains NP-complete on chordal graphs.

Paper organization. In Section 2 we explain our notations and terminology. In Section 3 we present a polynomial-time algorithm for solving the 2-Role Assignment problem for chordal graphs. In Section 4 we show that the k-Role Assignment problem for chordal graphs stays NP-complete for any fixed $k \geq 3$. Section 5 contains the conclusions and mentions some open problems.

2 Preliminaries

All graphs considered in this paper are undirected, finite and simple, i.e., without loops or multiple edges, unless stated otherwise. For terminology not defined below, we refer to [7].

Let $G = (V, E)$ be a chordal graph. A *clique tree* of G is a tree $T = (\mathcal{K}, \mathcal{E})$ such that $\mathcal{K}$ is the set of maximal cliques of G, and for each vertex $v \in V$ the set $\mathcal{K}_v$ of maximal cliques of G containing v induces a connected subtree in T. It is well-known that a graph is chordal if and only if it has a clique tree, and that a clique tree of a chordal graph can be constructed in linear time (cf. [14]). We refer to a set $K \in \mathcal{K}$ as a *bag* of T. We define the notions *root bag*, *parent bag*, *child bag* and *leaf bag* of a clique tree similar to the notions root, parent, child and leaf of a 'normal' tree. If the bag $K_r \in \mathcal{K}$ is the root bag of the clique tree T of G, then we say that T is *rooted at* K_r. Every bag $K \neq K_r$ of the clique tree T has exactly one parent bag K'. We say that a vertex $v \in K$ is *given to the parent bag* K' if $v \in K \cap K'$, i.e., if v is both in the child bag K and in the parent bag K'. We say that vertex $v \in K$ *stays behind* if $v \in K \setminus K'$, i.e., if v is in the child bag K but not in the parent bag K'.

A *hypergraph* H is a pair $(Q, \mathcal{S})$ consisting of a set $Q = \{q_1, \dots, q_m\}$, called the *vertices* of H, and a set $\mathcal{S} = \{S_1, \dots, S_n\}$ of nonempty subsets of Q, called the *hyperedges* of H. With a hypergraph $H = (Q, \mathcal{S})$ we associate its *incidence graph* I, which is a bipartite graph with partition classes Q and $\mathcal{S}$, where for any $q \in Q, S \in \mathcal{S}$ we have $qS \in E(I)$ if and only if $q \in S$. A 2-*coloring* of a hypergraph $H = (Q, \mathcal{S})$ is a partition (Q_1, Q_2) of Q such that $Q_1 \cap S_j \neq \emptyset$ and $Q_2 \cap S_j \neq \emptyset$ for $1 \leq j \leq n$. A hypergraph H is called *nontrivial* if Q contains at least three vertices. The Hypergraph 2-Colorability problem asks whether a given (nontrivial) hypergraph has a 2-coloring. This problem, also known as Set Splitting, is NP-complete (cf. [15]).

3 The polynomial algorithm for 2-role assignments

In this section, we prove the following result.

Theorem 1. *The* 2-Role Assignment *problem is solvable in polynomial time for the class of chordal graphs.*

We will start by discussing the different 2-role assignments. Let G be a chordal graph. Following the notation of Sheng [20], the six different role graphs on two vertices are $R_1 = (\{1,2\}, \emptyset)$, $R_2 = (\{1,2\}, \{22\})$, $R_3 = (\{1,2\}, \{11,22\})$, $R_4 = (\{1,2\}, \{12\})$, $R_5 = (\{1,2\}, \{12,22\})$ and $R_6 = (\{1,2\}, \{11,12,22\})$.

If G contains at most one vertex, then G has no 2-role assignment. Suppose $|V_G| \geq 2$. If G only contains isolated vertices, then G has an R_1-role assignment. If G contains at least one isolated vertex and at least one component with at least two vertices, then G has an R_2-role assignment. If G is disconnected but does not have isolated vertices, then G has an R_3-role assignment.

Now, assume that G is connected and has at least two vertices. If G is bipartite, then G has an R_4-role assignment. If G is non-bipartite, then G has a 2-role assignment if and only if G has an R_5-role assignment or an R_6-role assignment.

We claim that we only have to check whether G has an R_5-role assignment. This is immediately clear if G has a vertex of degree 1, as such a vertex must be mapped to a role of degree 1 and R_6 does not have such a role. If G does not have any degree 1 vertices, we use the following result by Sheng [20].

Theorem 2 ([20]). *Let G be a chordal graph with at most one vertex of degree 1 and no isolated vertices. Then G has an R_5-role assignment.*

We now present a polynomial-time algorithm that solves the R_5-Role Assignment problem for chordal graphs. From the above, it is clear that this suffices to prove Theorem 1. We start by giving an outline of the algorithm.

Our algorithm takes as input a clique tree $T = (\mathcal{K}, \mathcal{E})$ of a chordal graph $G = (V, E)$. The algorithm outputs an R_5-role assignment of G, or outputs No if such a role assignment does not exist. The algorithm consists of two phases.

Phase 1. Decide whether or not G has an R_5-role assignment.

In Phase 1, the algorithm processes the maximal cliques of G in a "bottom-up" manner, starting with the leaf bags of T, and processing a bag only after all its child bags have been processed. When processing a bag K with parent K', labels are assigned to the vertices in $K \cap K'$ maintaining the following invariant.

Invariant 1 *Let V' be the set of vertices of G minus those in K and its descendants. If G is R_5-role assignable, then a partial solution on $G[V']$ can be extended to a solution on G if and only if it satisfies the constraints given by the labels of the vertices on $K \cap K'$.*

Each label $L(v)$ contains information about the possible roles that v can get in any R_5-role assignment of G as well as information about the possible roles of the neighbors of v. The possible labels for a vertex v in a bag K are:

$L(v) = 0$	: initial label for every vertex
$L(v) = 1^*$	: exactly one vertex with label 1^* in this bag must get role 1, all others must get role 2
$L(v) = 1$	: v must get role 1, no role restrictions for its neighbors
$L(v) = 2$	: v must get role 2, no role restrictions for its neighbors
$L(v) = 2_1$	: v must get role 2, and at least one neighbor must get role 1
$L(v) = 2_2$	: v must get role 2, and at least one neighbor must get role 2
$L(v) = 1\|2$	: v can get either role 1 or 2 without restrictions
$L(v) = 1\|2_1$	: v can get role 1, or v can get role 2 in which case at least one neighbor must get role 1
$L(v) = 1\|2_2$	: v can get role 1, or v can get role 2 in which case at least one neighbor must get role 2

Initially, $L(v) = 0$ for every $v \in V$. The label of a vertex can change several times: the arrows in Figure 1 represent all possible transitions between two labels. This figure will be clarified in detail later on. For now, we only note that there no arrows point downwards in Figure 1. This corresponds to the fact that labels in a higher level contain more information than labels in a lower level. For example, if a vertex v in bag K has a label $L(v) = 2_2$ and one of its neighbors in K gets label 2, then we change the label of v into $L(v) = 2$ before processing the parent bag of K. After all, label 2 contains more information than label 2_2, as label 2 contains the information that at least one neighbor of v will get role 2 in Phase 2. Labels changes such as these can also be applied to vertices not in $K \cap K'$, these serve to simplify the algorithm.

The algorithm outputs No if conflicting labels are assigned to vertices in K. The easiest example of this is when K has two vertices with label 1 from different child bags; 1 has no self loop in R_5. Once the algorithm has successfully processed all maximal cliques and no conflicting labels have been created, it concludes that G has an R_5-role assignment and produces such a role assignment in Phase 2.

Phase 2. Produce an R_5-role assignment of G.

An R_5-role assignment of G is constructed in a greedy way satisfying the constraints imposed by the labels. Since Invariant 1 holds, we eventually obtain an R_5-role assignment on G.

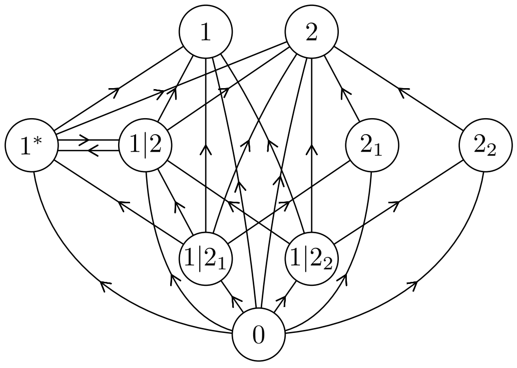

Fig. 1. All possible labels and all possible transitions between them.

Before we present the algorithm in detail, we make a useful observation.

Observation 1 *Let $T = (\mathcal{K}, \mathcal{E})$ be a clique tree of a connected chordal graph G, rooted at K_r. For any bag $K \neq K_r$ of T and its parent bag K', we have $K \cap K' \neq \emptyset$ and $K \setminus K' \neq \emptyset$. Moreover, for all K: $|K| \geq 2$.*

Theorem 3. *The R_5-*Role Assignment *problem is polynomial time solvable for the class of chordal graphs.*

Proof. Let G be a connected chordal graph, and let $T = (\mathcal{K}, \mathcal{E})$ be a clique tree of G rooted at K_r which we obtain in linear time (cf. [14]). We set $L(u) = 0$ for all $u \in V$ and start with Phase 1. Let K be the bag that is currently being processed. Recall that all child bags of K have already been processed. First assume $K \neq K_r$. So K has a parent bag K'. By Observation 1, at least one vertex in K stays behind, and at least one vertex is given to K'. We shall see that when our algorithm moves to K' then each vertex $u \in K \cap K'$ has $L(u) \neq 0$.

Suppose K is a leaf bag of T. Let v be the vertex that stays behind. If $|K| = 2$, then its other vertex x is given to K'. Because v has degree 1 in G, we must set $L(v) = 1$ and $L(x) = 2_2$, as v must get role 1, x must get role 2, and at least one other neighbor of x must get role 2. Suppose $|K| \geq 3$. We assign label $1|2$ to every vertex in K. We may do so because we can say that v *gives us the freedom to complete any assignment on* K. In other words, if in Phase 2 all vertices in $K \cap K'$ receive role 2, then we assign role 1 to v and role 2 to all remaining vertices of K. If one of the vertices in $K \cap K'$ receives role 1, then we assign role 2 to v and all remaining vertices of K.

We continue with *non-leaf bags* and consider several cases; each dealt with in polynomial time.

Case 1: K contains a v with $L(v) = 1$.
In any R_5-role assignment r of G with $r(v) = 1$, all vertices of $K \setminus \{v\}$ have role 2. Thus, if two vertices (coming from different child bags) in K have label 1, or other vertices in K have label 1^*, we have conflicting labels and output NO.

If $|K| \geq 3$, then we may assign label 2 to every vertex in $K \setminus \{v\}$; as at least two vertices get role 2, all vertices in $K \setminus \{v\}$ will have both a neighbor with role 1 (namely v) and a neighbor with role 2. If $|K| = 2$, then for the only vertex $x \in K \setminus \{v\}$ set $L(x) := 2$ if x had a label in $\{1|2, 1|2_1, 2, 2_1\}$ and set $L(x) := 2_2$ if x had a label in $\{0, 1|2_2, 2_2\}$.

Case 2: K contains a v with $L(v) = 1^*$ and $L(K) = \{0, 1^*, 1|2, 1|2_1, 1|2_2, 2, 2_1, 2_2\}$. If multiple sets of vertices with label 1^* originate from different child bags, we have conflicting labels and output No. Let $V^* := \{x \in K \mid L(x) = 1^*\}$. We assume that $|V^*| \geq 2$; otherwise replace label 1^* with label 1 and return to Case 1. Because of the vertices with label 1^*, K contains a vertex with role 1 and at least one vertex with role 2. Therefore, we may set $L(x) := 2$ for every $x \in K \setminus V^*$. If $V^* \subseteq (K \cap K')$, then we are done. Otherwise, at least one vertex $v \in V^*$ is left behind. This vertex gives us the freedom to complete any assignment to the vertices in $V^* \cap K'$. Hence, we relabel them to $1|2$. This is so, since if a vertex $x \in V^* \cap K'$ receives role 1, then all neighbors of x (including v) must receive role 2. If all vertices in $V^* \cap K'$ receive role 2, then we can give role 1 to v.

Case 3: K contains a v with $L(v) = 2_2$ and $L(K) = \{0, 1|2, 1|2_1, 1|2_2, 2, 2_1, 2_2\}$. If K contains any other vertex with label 2, 2_1 or 2_2, then v has its required neighbor and we replace the label 2_2 by 2. Furthermore, if $|K| \geq 3$, then at least one vertex in K will get role 2 and we can also replace the label 2_2 by 2. In both cases, we arrive at another case in this proof which we discuss later on.

The case where $|K| = 2$ remains; let x be the other vertex in K. If $L(x) = 1|2_2$, then we can replace it by $L(x) = 1|2$ since neighbor v will receive role 2. We conclude that by this argument and previous cases $L(x) \in \{0, 1|2, 1|2_1\}$. Now, either x stays behind, or x is given to K'.

If x stays behind, then v is given to K'. The label of x can only be influenced by v. If $L(x) = 1|2$, then x can function as the neighbor with role 2 that v needs, so we may set $L(v) = 2$. If $L(x) \in \{0, 1|2_1\}$, then the fact that none of the neighbors of x will receive role 1 (otherwise x would not have label 0 or $1|2_1$) means we must give x role 1 and leave $L(v) = 2_2$ unaltered. If x is given to K', then v stays behind while it still needs a neighbor with role 2, which can only be x. Hence, we apply the following relabelling to x that forces x to get role 2: $0 \rightarrow 2_1$, $1|2 \rightarrow 2$, $1|2_1 \rightarrow 2_1$.

Case 4: K contains a v with $L(v) = 2_1$ and $L(K) = \{0, 1|2, 1|2_1, 1|2_2, 2, 2_1\}$. Similar to the previous case, we can replace any label $1|2_2$ in K by $1|2$. Hence, by the above cases, we may assume that $L(x) \in \{0, 1|2, 1|2_1, 2, 2_1\}$ for every $x \in K$.

Suppose there exists a vertex $x \in K \setminus \{v\}$ with $L(x) \in \{0, 1|2, 1|2_1\}$ that stays behind. This vertex gives us the freedom to change the labels of every vertex $v' \in K \cap K'$ as follows: if $L(v') \in \{0, 1|2, 1|2_1\}$ then we set $L(v') := 1|2$, if $L(v') \in \{2, 2_1\}$ then we set $L(v') := 2$. This is so, since if none of the vertices in $K \cap K'$ receives role 1 in Phase 2, then Phase 2 assigns role 1 to x; otherwise x gets role 2. The latter is fine since v will also receive role 2.

Suppose $L(x) = 2$ for every vertex x that stays behind. For every vertex $v' \in K \cap K'$ with $L(v') = 0$, we set $L(v) = 1|2_1$. We may do this because v'

either gets role 1, or gets role 2 in which case it needs at least one neighbor to get role 1. We leave all other labels unaltered.

In the remaining case, at least one vertex with label 2_1 stays behind; w.l.o.g. let this be v. If $K \cap K'$ does not contain a vertex x with $L(x) \in \{0, 1|2, 1|2_1\}$, then we obtain a contradiction (x will never have a neighbor with role 1) and the algorithm outputs No. If $K \cap K'$ contains exactly one vertex x with $L(x) \in \{0, 1|2, 1|2_1\}$, this is the only vertex that can be the role 1 neighbor of v, hence we set $L(x) := 1$. If $K \cap K'$ contains multiple vertices with a label from $\{0, 1|2, 1|2_1\}$, then we set them all to 1^* by the same reasoning. Furthermore, since there will now be a vertex with role 1, we replace any 2_1 by 2 and any $1|2_1$ by $1|2$.

Case 5: K contains a v with $L(v) = 2$ and $L(K) = \{0, 1|2, 1|2_1, 1|2_2, 2\}$.
Since v will receive role 2 and every vertex in K is a neighbor of v, we may change the label of every vertex $x \in K$ with $L(x) = 1|2_2$ or $L(x) = 0$ into $L(x) = 1|2$ or $L(x) = 1|2_1$, respectively. Because of this and the previous cases, we may assume that $L(x) \in \{1|2, 1|2_1, 2\}$ for every vertex $x \in K$. Suppose x is a vertex of K that stays behind with $L(x) \in \{1|2, 1|2_1\}$. Then, just as before, x gives us the freedom to change any label $1|2_1$ in $K \cap K'$ to label $1|2$; x will be the neighbor with role 1 if necessary. Otherwise, if $L(x) = 2$ for all $x \in K \setminus K'$, then we leave all labels in K unaltered; all vertices in $K \setminus K'$ have the required neighbors and add nothing useful to K'.

Case 6: $L(v) \in \{0, 1|2, 1|2_1, 1|2_2\}$ for every $v \in K$.

Case 6a: there exists an $x \in K$ with $L(x) = 1|2$ that is left behind.
If $|K| \geq 3$, then x gives us the freedom to complete any assignment on $K \cap K'$ and we set $L(v) = 1|2$ for all $v \in K$. This is true, because each vertex will get a neighbor with role 2 anyway ($|K| \geq 3$), and if no vertex of $K \cap K'$ gets role 1 in Phase 2, we give role 1 to x. Otherwise, $|K| = 2$; let $K = \{x, y\}$ where y is given to K'. If y will get role 1 in Phase 2, then x will get role 2. If y gets role 2, then it already has a neighbor in K' ($K \setminus K' \neq \emptyset$) with some role and we set x to have the other. Hence, we may set $L(y) := 1|2$.

Case 6b: there exists an $x \in K$ with $L(x) = 1|2_1$ that is left behind.
Suppose $|K| \geq 3$. Using the same arguments as in Case 6a, we can show that we may assign label $1|2$ to every vertex in $K \cap K'$. Otherwise $|K| = 2$. Let $K = \{x, y\}$ where y is given to K'. Notice that if we give y role 1, then we can complete the role assignment by giving role 2 to x. If we want to give role 2 to y, then x will get role 1, since otherwise it has no role 1 neighbor. In this case, the requirements on y depend on the current label of y. As a result, we apply the following replacements for the label of y: $0 \to 1|2_2$, $1|2 \to 1|2$, $1|2_1 \to 1|2$, $1|2_2 \to 1|2_2$.

Case 6c: there exists an x with $L(x) = 1|2_2$ that is left behind.
If $|K| \geq 3$, then at least one vertex in K gets role 2. Hence we may change the label of x into $1|2$ and return to Case 6a. Thus $|K| = 2$; let $K = \{x, y\}$ where y is given to K'. In this case, y can not get role 1 because then x must get role 2. This is not possible, as then x does not have a neighbor with role 2. We maintain

Invariant 1 as follows. If $L(y) \in \{1|2, 1|2_1\}$, then we set $L(y) := 2$. Then x can get role 1. If $L(y) = 1|2_2$, then set $L(y) := 2$ and x will receive role 2. If $L(y) = 0$, we also set $L(y) := 2$. In that case y needs two neighbors with different roles: in Phase 2 a neighbor of y in K' gets some role. Then we give x the other role.

Case 6d: there exists an x with $L(x) = 0$ that is left behind.
If $|K| \geq 3$ then at least one vertex in K gets role 2. Hence we may change $L(x)$ into $1|2_1$ and return to Case 6b. Thus $|K| = 2$; let $K = \{x, y\}$ where y is given to K'. As x has label 0 and is left behind, x must have degree one in G. Then it must get role 1 while y must get role 2. This leads to the following replacement rules for the label of y: $1|2 \to 2$, $1|2_1 \to 2$, $1|2_2 \to 2_2$. Note that $L(y) \neq 0$, because then we are in a leaf bag.

At some moment we arrive at root bag K_r. Note that we can check in polynomial time if K_r allows an assignment of roles 1 and 2 such that (i) no two vertices in K_r get role 1 and (ii) the labels of the vertices in K_r are satisfied. If we find such an assignment, then we are done by Invariant 1 (which has been maintained during Phase 1). We now start with Phase 2 of the algorithm which follows the reasoning used in Phase 1 in reverse order. $\square$

Remark. In our polynomial-time algorithm that solves the 2-Role Assignment problem for chordal graphs we do not have to check if the input graph has an R_6-role assignment (cf. Theorem 2). This is very "fortunate" as the R_6-Role Assignment problem remains NP-complete when restricted to chordal graphs. This can be seen as follows. Let $(Q, \mathcal{S})$ be a nontrivial hypergraph. In its incidence graph I we add an edge between every pair of vertices in Q. This results in a chordal graph G. It is easy to see that $(Q, \mathcal{S})$ has a 2-coloring if and only if G has an R_6-Role Assignment.

4 Complexity of the role assignment problem for $k \geq 3$

It is known that the k-Role Assignment problem is NP-complete for any fixed $k \geq 2$ [12]. Theorem 1 states the 2-Role Assignment becomes polynomially solvable when the instance graph is chordal. In this section, we show that the problem for chordal graphs gets NP-complete again when k jumps to 3. Our NP-completeness construction is more involved than the one for general graphs in [12] as the latter is not chordal.

Theorem 4. *For $k \geq 3$, the k-Role Assignment problem is NP-complete for the class of chordal graphs.*

Proof. Let $k \geq 3$. We use a reduction from Hypergraph 2-Coloring. Let $(Q, \mathcal{S})$ be a nontrivial hypergraph with incidence graph I.

We modify I as follows. Firstly, we add an edge between any two vertices in Q; so Q becomes a clique. Secondly, for each $S \in \mathcal{S}$ we take a path $P^S = p_1^S \cdots p_{k-2}^S$ and connect it to S by the edge $p_{k-2}^S S$; so these new paths P^S are pendant paths in the resulting graph. Thirdly, we add a copy H^q of a new graph H for each

$q \in Q$. Before we explain how to do this, we first define H. Start with a path $u_1 u_2 \cdots u_{2k-4}$. Then take a complete graph on four vertices a, b, c, d, and a complete graph on four vertices w, x, y, z. Add the edges $cu_1, du_1, u_{2k-4}w, u_{2k-4}x$. We then take three paths $S = s_1 \cdots s_{k-2}$, $T = t_1 \cdots t_{k-2}$ and $T' = t'_1 \cdots t'_{k-2}$, and we add the edges $s_{k-2}w, ct_{k-2}, dt'_{k-2}$. This finishes the construction of H. We connect a copy H^q to q via the edge qu_1^q, where u_1^q is the copy of the vertex u_1. We call the resulting graph G; notice that this is a connected chordal graph. See Figure 2 for an example.

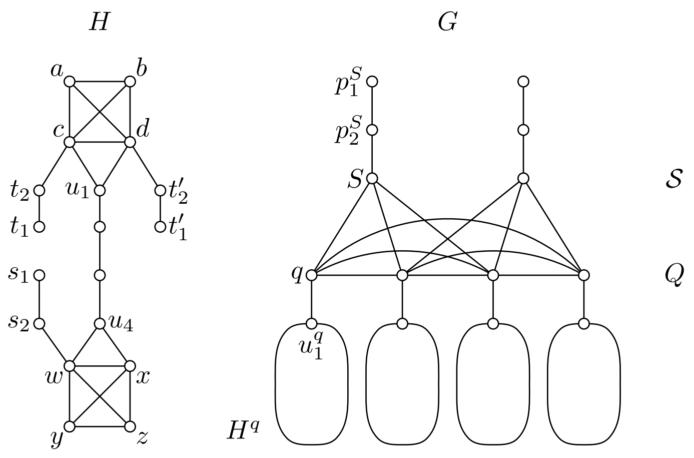

Fig. 2. The graph H (left side) and the graph G (right side) when $k = 4$.

We first show that if G has a k-role assignment r, then it has an R^*-role assignment, where R^* denotes the path $r_1 \cdots r_k$ on k vertices with a self-loop in vertices r_{k-1} and r_k. To see this, consider a copy H^q of H in G; we show that we can assign roles to the vertices of H^q in only one way. For convenience, we denote the vertices of H^q without the superscript q.

Note that r must map an induced path of length smaller than k in G to an induced path of the same length in R. Otherwise, r is not a k-role assignment. Hence, we may write $r(t_i) = i$ for $i = 1, \ldots, k-2$ and $r(c) = k-1$. This implies that a vertex with role 1 only has vertices with role 2 in its neighborhood and a vertex with role i for $2 \leq i \leq k-2$ only has vertices with role $i-1$ and role $i+1$ as neighbors. Then a vertex with role k can only be adjacent to vertices with role $k-1$ or role k. Hence c must have a neighbor with role k.

Suppose $r(d) = k$. Then $r(t'_{k-2}) \in \{k-1, k\}$ and this eventually leads to $r(t'_1) \geq 2$ without a neighbor of role $r(t'_1) - 1$ for t'_1. This is not possible. Hence $r(d) \neq k$. This means that $k \in r(\{a, b, u_1\})$. Since a, b, u_1 are neighbors of d as well and a vertex with role k can only have neighbors with role $k-1$ and k, we then find that d has role $k-1$.

The above implies that a and b have their role in $\{k-2, k-1, k\}$. Suppose $k = 3$. If $r(a) = 1$, then $r(b) = 2$ implying that r is a 2-role assignment (as $r(c) = r(d) = 2$). Suppose $r(a) = 2$. Then a needs a neighbor with role 1. Hence $r(b) = 1$, but then r is a 2-role assignment. Suppose $r(a) = 3$. Then $r(b) \neq 2$, as otherwise b needs a neighbor with role 1. Hence $r(b) = 3$. This means that r is an R^*-role assignment. Suppose $k \geq 4$. If $r(a) = k-2$, then a needs a neighbor with role $k-3$. So, $r(b) = k-3$. However, this is not possible since vertex b with role $k-3$ is adjacent to vertex c with role $k-1$. If $r(a) = k-1$, then $r(b) = k-2$. This is not possible either. Hence $r(a) = k$ and for the same reasons $r(b) = k$. Then r is an R^*-role assignment.

We claim that $(Q, \mathcal{S})$ has a 2-coloring if and only if G has a k-role assignment.

Suppose $(Q, \mathcal{S})$ has a 2-coloring (Q_1, Q_2). We show that G has an R^*-role assignment, which is a k-role assignment. We assign role i to each p_i^S for $i = 1, \ldots, k-2$ and role $k-1$ to each $S \in \mathcal{S}$. As $(Q, \mathcal{S})$ is nontrivial, either Q_1 or Q_2, say Q_2, has size at least two. Then we assign role $k-1$ to each $q \in Q_1$ and role $k-2$ to neighbor u_1^q. We assign role k to each $q \in Q_2$ and $k-1$ to neighbor u_1^q. As $|Q_2| \geq 2$, every vertex in Q has a neighbor with role k. Hence, we can finish off the role assignment by assigning roles to the remaining vertices of each copy H^q of H as follows. For convenience, we remove the superscript q. We map each path S, T, T' to the path $1 \cdots k-2$, where $r(s_i) = r(t_i) = r(t'_i) = i$ for $i = 1, \ldots, k-2$. If u_1 received role $k-2$ we assign u_i role $k-1-i$ for $i = 2, \ldots, k-2$ and we assign u_{k-2+i} role $i+1$ for $i = 1, \ldots, k-2$. Furthermore, we assign role $k-1$ to c, d, w, and role k to a, b, x, y, z. If u_1 received role $k-1$, it already has a neighbor with role k (namely its neighbor in Q). Then we assign u_i role $k-i$ for $i = 2, \ldots, k-1$ and we assign u_{k-1+i} role $i+1$ for $i = 1, \ldots, k-3$. Furthermore, we assign role $k-1$ to c, d, w, x, and role k to a, b, y, z.

To prove the reverse statement, suppose G has a k-role assignment r. As we have shown above, by construction, G must have an R^*-role assignment. Then each p_i^S must have role i for $i = 1, \ldots, k-2$. Then $r(S) = k-1$ for each $S \in \mathcal{S}$, and each S must have a neighbor in Q with role $k-1$ and a neighbor in Q with role k. We define $Q_1 = \{q \in Q \mid r(q) = k-1\}$ and $Q_2 = Q \backslash Q_1$. Then we find that (Q_1, Q_2) is a 2-coloring of $(Q, \mathcal{S})$. This completes the proof of Theorem 4. $\square$

5 Conclusions

We have settled an open problem of Sheng [20] by showing that it can be decided in polynomial time if a chordal graph has a k-role assignment when $k = 2$. We also showed that for any fixed $k \geq 3$ the problem stays NP-complete when restricted to chordal graphs.

Role assignments are also studied in topological graph theory. There, a graph G is called an *emulator* of a graph R if G has an R-role assignment. Then the question is which graphs allow planar emulators, see e.g. the recent manuscript [18] for nice developments in this area. An interesting question is the computational complexity of the k-Role Assignment problem for planar graphs. The answer to this question is already unknown for $k = 2$.

Acknowledgements. The second author thanks Jiří Fiala for fruitful discussions on the subject.

References

1. Abello, J., Fellows, M. R., and Stillwell, J. C. On the complexity and combinatorics of covering finite complexes. *Australian Journal of Combinatorics 4* (1991) 103–112.
2. Angluin, D. Local and global properties in networks of processors. In *12th ACM Symposium on Theory of Computing* (1980) 82–93.
3. Angluin, D., and Gardiner, A. Finite common coverings of pairs of regular graphs. *Journal of Combinatorial Theory B 30* (1981) 184–187.
4. Biggs, N. Constructing 5-arc transitive cubic graphs. *Journal of London Mathematical Society II. 26* (1982) 193–200.
5. Bodlaender, H. L. The classification of coverings of processor networks. *Journal of Parallel Distributed Computing 6* (1989) 166–182.
6. Chalopin, J., Métivier, Y., and Zielonka, W. Local computations in graphs: the case of cellular edge local computations. *Fundamenta Informaticae 74* (2006) 85–114.
7. R. Diestel. *Graph Theory.* (3rd Edition). Springer-Verlag Heidelberg, 2005.
8. Everett, M. G., and Borgatti, S. Role colouring a graph. *Mathematical Social Sciences 21* (1991) 183–188.
9. Fiala, J., and Kratochvíl, J. Complexity of partial covers of graphs. In *12th International Symposium on Algorithms and Computation (ISAAC 2001)* Lecture Notes in Computer Science 2223 (2001) 537–549.
10. Fiala, J., and Kratochvíl, J. Partial covers of graphs. *Discussiones Mathematicae Graph Theory 22* (2002) 89–99.
11. Fiala, J., Kratochvíl, J., and Kloks, T. Fixed-parameter complexity of λ-labelings. *Discrete Applied Mathematics 113* (2001) 59–72.
12. Fiala, J., and Paulusma, D. A complete complexity classification of the role assignment problem. *Theoretical Computer Science 349* (2005) 67–81.
13. Fiala, J., and Paulusma, D. Comparing universal covers in polynomial time. *Theory of Computing Systems*, to appear.
14. Galinier, P., Habib, M., and Paul, C. Chordal graphs and their clique graphs. In: Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1995), *Lecture Notes in Computer Science 1017* (1995) 358–371.
15. Garey, M. R., and Johnson, D. S. *Computers and Intractability.* W. H. Freeman and Co., New York, 1979.
16. Kratochvíl, J., Proskurowski, A., and Telle, J. A. Covering regular graphs. *Journal of Combinatorial Theory B 71* (1997) 1–16.
17. Pekeč, A., and Roberts, F. S. The role assignment model nearly fits most social networks. *Mathematical Social Sciences 41* (2001) 275–293.
18. Rieck, Y., and Yamashita, Y. Finite planar emulators for $K_{4,5} - 4K_2$ and Fellows' conjecture. manuscript (2009) arXiv:0812.3700v2.
19. Roberts, F. S., and Sheng, L. How hard is it to determine if a graph has a 2-role assignment? *Networks 37* (2001) 67–73.
20. Sheng, L. 2-Role assignments on triangulated graphs. *Theoretical Computer Science 304* (2003) 201–214.