SIAM J. Discrete Math.
Vol. 28, No. 4, pp. 1675–1685

ON LIST COLORING AND LIST HOMOMORPHISM OF PERMUTATION AND INTERVAL GRAPHS*

JESSICA ENRIGHT†, LORNA STEWART‡, AND GÁBOR TARDOS§

Abstract. List coloring is an NP-complete decision problem even if the total number of colors is three. It is hard even on planar bipartite graphs. We give a polynomial-time algorithm for solving list coloring of permutation graphs with a bounded total number of colors. More generally, we give a polynomial-time algorithm that solves the list-homomorphism problem to any fixed target graph for a large class of input graphs, including all permutation and interval graphs.

Key words. homomorphism, interval graph, permutation graph, list coloring

AMS subject classifications. 05C15, 68R10, 05C85

DOI. 10.1137/13090465X

1. Introduction.

A *proper coloring* of a graph assigns colors to the vertices such that adjacent vertices receive distinct colors. (In this paper we deal only with vertex colorings.) The k-coloring problem asks whether a given graph has a proper coloring with at most k colors. For $k \geq 3$ this is NP-complete.

In the *list coloring problem*, each vertex of the input graph comes with a list of allowed colors and we ask whether a proper coloring exists where each vertex receives a color from its list. As a generalization of ordinary coloring, it is NP-complete [10]. List coloring remains hard even on interval graphs [1], as well as split graphs, cographs, and bipartite graphs [9]. It is solvable in polynomial time on graphs of fixed treewidth [9].

Kratochvíl and Tuza [11] showed that list coloring is NP-complete even if the size of each list assigned to a vertex is at most three, each color appears in at most three lists, each vertex in the graph has degree at most three, and the graph is planar. However, they gave polynomial-time algorithms to solve list coloring on a graph if the maximum list size is at most two, or each color appears in at most two lists, or each vertex has degree at most two.

Let *k-list coloring* stand for the list coloring problem where the total number of colors is bounded by the constant k. This is a generalization of k-coloring, and thus for $k \geq 3$ it is NP-complete. It remains NP-complete on planar bipartite graphs [11], but is solvable in polynomial time on cographs [9], P_5-free graphs [8], and graphs of fixed treewidth [9]. See [6] for several related results, including a polynomial-time algorithm for k-list coloring for a graph class that the paper calls *treed graphs*, which contains cographs.

*Received by the editors January 3, 2013; accepted for publication (in revised form) July 7, 2014; published electronically October 9, 2014.
http://www.siam.org/journals/sidma/28-4/90465.html

†University of Glasgow, Glasgow G12 8QQ, UK (jessica.enright@glasgow.ac.uk). This author's research was supported by an iCORE ICT Graduate Student Scholarship and a University of Alberta Dissertation Fellowship.

‡Department of Computing Science, University of Alberta, Edmonton T6G 2E8, Canada (lorna.stewart@ualberta.ca). This author's research was supported by an NSERC Discovery grant.

§Alfred Rényi Institute of Mathematics, Rényi Institute, H-1053 Budapest, Hungary (tardos@renyi.hu). This author's research was supported by an NSERC Discovery grant, grant OTKA NN-102029, and the "Lendület" project of the Hungarian Academy of Sciences.

Note that 2-list coloring is solvable in polynomial time. Indeed, 2-coloring is solvable in polynomial time and has at most two (complementary) solutions on each connected component. Thus, for the 2-list coloring problem it is enough to check that one of these is compatible with the lists on each component.

A *graph homomorphism* from a graph G to another graph H is a function $f : V(G) \to V(H)$ satisfying that $f(x)$ and $f(y)$ are adjacent in H whenever x and y are adjacent in G. Note that here we allow the graphs to have loops.

Let H be a fixed graph. The H-coloring problem takes a graph G as input and asks whether there is a G to H homomorphism. In the list H-coloring problem, each vertex of the input graph comes with a list of vertices of H and we ask whether a G to H homomorphism exists that maps each vertex to a member of its list. Clearly, k-coloring is a graph homomorphism to the complete graph K_k, so list H-coloring is a generalization of k-list coloring. Polynomial-time algorithms are known for H-coloring [14] and list H-coloring [5] for graphs of fixed treewidth. The complexities of the H-coloring and list H-coloring problems for arbitrary input graphs are completely characterized in terms of the structure of H [7].

Permutation graphs are comparability cocomparability graphs (see definitions in the next section). List coloring is NP-complete on permutation graphs since cographs are permutation graphs [9]. The k-list coloring problem is NP-complete for comparability graphs for $k \geq 3$ since bipartite graphs are comparability graphs. The complexity of k-list coloring of cocomparability graphs remains open.

In this paper we give a polynomial-time algorithm for the k-list coloring of permutation graphs for any fixed k. More generally, we give a polynomial-time algorithm that solves the list-homomorphism problem to any fixed target graph for permutation graphs. The same algorithm also works for interval graphs and more.

Subsequent to this work, Valadkhan [15] gave another polynomial-time algorithm for k-list coloring (or, more generally, the list-homomorphism problem) for permutation and interval graphs. His approach is somewhat simpler and has a lower exponent in the polynomial running time, but it applies only to interval and permutation graphs.

Our algorithm is based on what we call a *multichain ordering* (see definition in the next section), a notion related to chain graphs [16] and to a characterization of bipartite permutation graphs given in [4]. The algorithm applies to every graph with all connected induced subgraphs having a multichain ordering, among them all permutation graphs and all interval graphs. We also remark that since adding loops to a graph does not have any effect on the multichain ordering, our algorithm also applies to interval and permutation graphs with loops added to some vertices. The running time for k-list coloring, or, more generally, for list H-coloring for a graph H on k vertices, is $O(n^{k^2-3k+4})$, where n stands for the number of vertices of the input graph.

Hoàng et al. [8] give an algorithm for k-list coloring P_5-free graphs in polynomial time. Their algorithm, like ours, is based on how coloring of one side of the bipartition of a chain graph can restrict the coloring of the other side and noticing that there are only a polynomial number of possible such restrictions.

We mention here that a polynomial-time k-list coloring algorithm for interval graphs cannot be considered new. Indeed, as previously mentioned, a polynomial-time algorithm already exists for list coloring graphs with bounded treewidth. The treewidth of an interval graph is one less than the size of its largest clique. Thus, unless it is bounded, one does not have a proper coloring with a bounded number of colors. The same cannot be said about permutation graphs though. Even bipartite permutation graphs have unbounded treewidth.

Multichain orderings are based on distance from a starting vertex. They give insight into the structure of permutation or interval graphs and may lead to algorithms for other problems on these or similar graphs.

2. Definitions and preliminaries.

We consider only finite graphs with no multiple edges. We allow for loop edges connecting a vertex to itself and call a graph *simple* if it has no such edge. (Loop edges in the input graph make sense for the list H-coloring problem only if H has at least one loop, so, in particular, it does not make sense for k-list coloring.) We represent graphs as a pair $G = (V, E)$, where $V = V(G)$ is the vertex set and $E = E(G)$ is the edge set. We denote the edge connecting x to y by xy, so $xy = yx$. In a *directed graph* we have ordered pairs of vertices as edges and denote such an edge as $\overrightarrow{uv}$, saying it leaves the vertex u and is oriented toward the vertex v. A *sink* is a vertex that no edge leaves. A directed graph is *transitive* if the presence of the edges $\overrightarrow{uv}$ and $\overrightarrow{vw}$ implies the presence of $\overrightarrow{uw}$. An *orientation* of the simple graph $G = (V, E)$ is a directed graph $G = (V, \overrightarrow{E})$, where $\overrightarrow{E}$ is obtained by replacing each edge $uv \in E$ by one of its orientations: $\overrightarrow{uv}$ or $\overrightarrow{vu}$ but not both. A *comparability graph* is a simple graph that admits a transitive orientation. Equivalently, a graph is a comparability graph if there is a partial order on the vertices with exactly the adjacent (distinct) vertices being comparable. The *complement* of the simple graph $G = (V, E)$ is $\overline{G} = (V, \overline{E})$, where $\overline{E}$ contains all possible nonloop edges on V not in E. We sometimes call the edges of $\overline{G}$ the *nonedges* of G. A *cocomparability graph* is a graph whose complement is a comparability graph. Graphs that are simultaneously comparability and cocomparability graphs are known as *permutation graphs*. Permutation graphs are exactly the graphs $G = (V, E)$ that are obtained from a permutation $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ by setting $V = \{x_1, \dots, x_n\}$ and $E = \{x_i x_j \mid i < j, \pi(i) > \pi(j)\}$. A simple graph is an *interval graph* if one can identify its vertices with real intervals such that two vertices are adjacent if and only if the corresponding intervals intersect. Such intervals can always be chosen to have distinct endpoints. The graph C_n is the cycle with n vertices and n edges. *Weakly chordal graphs* are simple graphs with no induced C_n or $\overline{C_n}$ for $n > 4$. Three vertices form an *asteroidal triple* in a graph if, between each pair, there is a path that avoids the neighborhood of the third vertex. A graph is *asteroidal triple–free* or *AT-free* if it has no asteroidal triple. For further information about these graph classes and other graphs mentioned in this paper, the reader is referred to [3].

Let $G = (V, E)$ be a graph. A *list mapping* of G is a mapping that assigns a set (list) of colors to each vertex in G. A coloring of G *obeys* a list mapping if it assigns every vertex a color from its list. More generally, if the graph H is fixed, a *list mapping* of G assigns a subset of $V(H)$ (a list) to every vertex of G. A homomorphism from G to H *obeys* the list mapping if each vertex is mapped to a member of its list.

A *chain graph* is a bipartite graph that contains no induced $2K_2$ (the complement of C_4). This name was introduced by Yannakakis [16]. The following characterization is easily seen to be equivalent to the definition. A bipartite graph with sides (partite sets) A and B is a chain graph if and only if for any two vertices in A the neighborhood of one of them contains the neighborhood of the other. As a consequence we see that if we order the vertices of B according to decreasing degree (breaking ties arbitrarily), then the neighborhood of any vertex in A is an initial segment in this ordering.

Let $G = (V, E)$ be a connected graph. The *distance layers of* G from a vertex v_0 are $\{v_0\} = L_0, L_1, \dots, L_z$, where L_i consists of the vertices at distance i from v_0 and z is the largest integer for which this set is not empty. These layers form a *multichain*

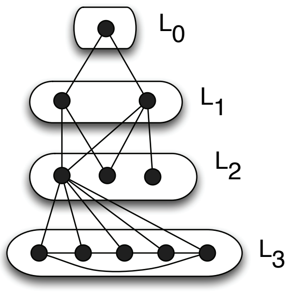

FIG. 1. *A graph with a multichain ordering as shown. The subgraph induced by the vertices of L_3 does not have a multichain ordering.*

ordering of G if for every two consecutive layers L_i and L_{i+1} the edges connecting these two layers form a chain graph. Note that a graph that has a multichain ordering may contain an induced subgraph that does not have a multichain ordering. For example, see Figure 1.

THEOREM 2.1. *Let H be a fixed graph. The H-coloring problem is polynomial-time solvable for input graphs G satisfying that every connected induced subgraph of G admits a multichain ordering.*

We prove this theorem in the next section by presenting an algorithm, proving its correctness, and estimating its running time. Our algorithm processes each connected component of the input graph separately. It is based on multichain orderings of the components and uses the following simple properties of such orderings: (a) H-coloring of one layer in a multichain ordering has a limited effect on the coloring of the next layer and no direct effect on subsequent layers, and (b) each layer has a vertex that is adjacent to all vertices in the next layer; thus if this vertex is mapped to c, then all nonneighbors of c will be missing from the H-coloring of the next layer, practically reducing the size of H. Note that (b) does not apply if H has a vertex c that is adjacent to every vertex of H including itself. Fortunately this easy special case can be handled by alternate methods.

In the remainder of this section we show that permutation and interval graphs do satisfy the requirement of the theorem. We also show that there are graphs beyond these two categories that also satisfy the same requirements.

LEMMA 2.2. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a transitive orientation of a connected comparability graph $G = (V, E)$. Let $v_0 \in V$ be a sink in $\overrightarrow{G}$, and let $L_0, \ldots, L_z$ be the distance layers of G from v_0. Then for $0 \leq i < z$ all edges of $\overrightarrow{E}$ between the vertices of two consecutive layers L_i and L_{i+1} are oriented toward L_i if i is even and all these edges are oriented toward L_{i+1} if i is odd.*

Proof. We proceed by induction on i. For $i = 0$ the statement of the lemma holds since v_0 is a sink. Each $u \in L_i$ for $i > 0$ has a neighbor $u' \in L_{i-1}$, and an edge between u and L_{i+1} oriented "the wrong way" would imply the presence of an edge between u' and L_{i+1} by transitivity, a contradiction. $\square$

LEMMA 2.3. *Let $\overrightarrow{\overline{G}} = (V, \overrightarrow{\overline{E}})$ be a transitive orientation of the complement of a connected cocomparability graph $G = (V, E)$. Let $v_0 \in V$ be a sink in $\overrightarrow{\overline{G}}$, and let $L_0, \ldots, L_z$ be the distance layers of G from the vertex v_0. Then for every pair of layers L_i, L_j where $0 \leq i < j \leq z$ all edges of $\overrightarrow{\overline{G}}$ between L_i and L_j are directed toward L_i.*

Proof. We proceed by induction on i. For $i = 0$ the statement follows from v_0 being a sink. Let us consider $i > 0$ and assume for contradiction that $\overrightarrow{uv}$ is an edge of $\overset{\rightrightarrows}{G}$ with $u \in L_i$ and $v \in L_j$, $j > i$. Now v is not adjacent in G to any vertex $u' \in L_{i-1}$, so by the induction hypothesis we have $\overrightarrow{vu'} \in \overset{\rightrightarrows}{E}$ and by transitivity $\overrightarrow{uu'} \in \overset{\rightrightarrows}{E}$. But this contradicts the fact that u has a neighbor $u' \in L_{i-1}$ in G, so no orientation of an edge between u and this neighbor should be present in $\overset{\rightrightarrows}{G}$. □

THEOREM 2.4. *Every connected permutation graph has a multichain ordering.*

Proof. Let $G = (V, E)$ be a permutation graph, and let $\overrightarrow{G}$ be a transitive orientation of G and $\overset{\rightrightarrows}{G}$ a transitive orientation of the complement of G.

Let v_0 be a vertex that is a sink in both of the graphs $\overrightarrow{G}$ and $\overset{\rightrightarrows}{G}$, the existence of which is shown in [13]. We claim that the distance layers $L_0, \dots, L_z$ of G from v_0 form a multichain ordering. To see this assume for a contradiction that $u, v \in L_i$ and $u', v' \in L_{i-1}$ are vertices of two neighboring layers and u is adjacent with u' but not with v' in G and v is adjacent with v' but not with u'. We distinguish two cases according to whether u and v are adjacent in G.

Assume first that u and v are adjacent, and assume without loss of generality that this edge is oriented toward v in $\overrightarrow{G}$. By Lemma 2.2, either both the edge between u and u' and that between v and v' are oriented toward L_i, or both are oriented toward L_{i-1}. In the former case we should have $\overrightarrow{u'v} \in \overrightarrow{E}$ by transitivity, contradicting our assumption that v is not adjacent with u' in G. In the latter case we similarly have $\overrightarrow{uv'} \in \overrightarrow{E}$, again a contradiction.

Now assume that u and v are not adjacent in G, and assume without loss of generality that this nonedge is oriented toward v in $\overset{\rightrightarrows}{G}$. By Lemma 2.3, the nonedge between v and u' is oriented toward u' in $\overset{\rightrightarrows}{G}$. By transitivity we have $\overrightarrow{uu'} \in \overset{\rightrightarrows}{E}$, contradicting our assumption that u and u' are adjacent in G. □

Not every graph with a multichain ordering is a permutation graph. Further examples are given by interval graphs, as shown by the next theorem. In addition, C_n and $\overline{C_n}$, where $n > 4$, and the graph T defined as the star $K_{1,3}$ with each edge subdivided once do not have multichain orderings. Therefore, there are cocomparability graphs, weakly chordal graphs, and even trees that do not have multichain orderings. Moreover, any graph such that all induced subgraphs have multichain orderings must be a weakly chordal graph. We also note that each connected induced subgraph of the complement of any forest admits a multichain ordering (start with a vertex of largest degree, and L_1 will be the only layer with more than a single vertex). These complements-of-forests are cocomparability graphs, but some of them, including the complement of the tree T, are neither permutation nor interval graphs. The graph of Figure 2 gives an example of a graph that contains an asteroidal triple and still has the property that every connected induced subgraph has a multichain ordering.

THEOREM 2.5. *All connected interval graphs admit multichain orderings.*

Proof. Consider an interval representation in which all interval endpoints are distinct. We can choose v_0 to be the vertex with the leftmost left endpoint. One can find reals $x_0 < x_1 \dots$ such that the layer L_i of G at distance i from v_0 consists of the vertices with their left endpoint in $(x_{i-1}, x_i]$. To see that these layers form a multichain ordering of G, take two vertices (intervals) in L_i and let u be the one with its left endpoint more to the left, and v the other. Clearly, all intervals in L_{i-1} intersecting v must also intersect u. □

Our list H-coloring algorithm works for every graph whose connected induced

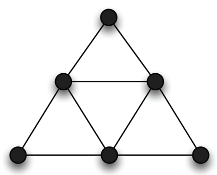

FIG. 2. *The 3-sun contains an asteroidal triple, and every connected induced subgraph has a multichain ordering.*

subgraphs all have multichain orderings, and it runs in polynomial time as long as H is fixed. The last two theorems show that this class includes all permutation and interval graphs (and the graphs obtained from them by adding some loops). Restricting our attention to complete graphs $H = K_k$ we get polynomial-time k-list coloring algorithms.

Given a connected graph G and vertex v of G, we can check whether the distance layers from starting vertex v form a multichain ordering in $O(m)$ time, where m is the number of edges of G. The algorithm for doing so uses breadth-first search to generate the distance layers from v. It then uses bucket sort to order the vertices of each layer by decreasing the size of their neighborhood in the previous layer. Finally it checks that for each vertex it holds that its neighbors in the next layer appear in the beginning of that layer before the nonneighbors. Each step can be accomplished in $O(m)$ time.

As a naive algorithm to check whether a connected graph has a multichain ordering, and to generate it if it does, we can start a breadth-first search from each vertex and check to see whether that search has given us a multichain ordering in $O(nm)$ time overall. In some classes, for example, permutation graphs, this can be done more quickly. In the case of permutation graphs, we can use the output of the linear-time recognition algorithm provided by McConnell and Spinrad [12] to identify a vertex that is a sink in some transitive orientation of both the graph and its complement. We can then generate the distance layers from this vertex in $O(m)$ time, which is a multichain ordering, as the proof of Theorem 2.4 shows. Similarly, several linear-time algorithms exist to find a "leftmost" vertex in a interval graph, the earliest one being in [2]. The distance layers can be constructed from there in linear time. As the proof of Theorem 2.5 shows, this is a multichain ordering.

3. The algorithm. In this section we present our algorithm to list H-color any graph with the property that all connected induced subgraphs have multichain orderings. The algorithm runs in polynomial time if H is fixed. Since the algorithm handles connected components separately, we consider only connected graphs in the following description.

Let $G = (V, E)$ be a connected graph, and let $L_0, \ldots, L_z$ form a multichain ordering of G. For $x \in L_i$ we introduce $d_-(x)$ for the number of neighbors of x in L_{i-1} (or 0 if $i = 0$) and $d_+(x)$ for the number of neighbors of x in L_{i+1} (or 0 if $i = z$). We fix an ordering of the vertices within each layer according to decreasing d_- values, breaking ties arbitrarily. As observed in the definition of chain graphs, this ordering ensures that the neighbors of a vertex $x \in L_i$ among the vertices of the next layer L_{i+1} must be the first $d_+(x)$ vertices in that layer.

Let us fix the target graph H with vertex set $C = V(H)$. Let $\mathcal{P}$ be a list mapping of G, so $\mathcal{P}(x) \subseteq C$ for every vertex $x \in V$.

A *configuration* is a pair (i, B), where $1 \leq i \leq z$ and $B : C \to \{0, 1, \ldots, |L_i|\}$, satisfying that B takes both 0 and $|L_i|$ as values. We introduce two more special configurations: $S_0 = (0, B_0)$ and $S_{z+1} = (z + 1, B_0)$, where $B_0 : C \to \{0\}$ is the constant zero function.

These configurations form the vertices of the *configuration graph*. This is a directed graph that contains the edge from (i, B) to (i', B') if $i' = i + 1$, and there is a homomorphism χ from the subgraph G_i of G induced by the layer L_i to H *providing* for this edge, i.e., satisfying the following three conditions:

1. χ obeys $\mathcal{P}$, i.e., for $x \in L_i$ we have $\chi(x) \in \mathcal{P}(x)$.
2. χ does not assign $c \in C$ to the first $B(c)$ vertices in L_i (recall that L_i is ordered).
3. For each $x \in L_i$ and $c \in C$ with c not adjacent to $\chi(x)$ in H we have $B'(c) \geq d_+(x)$.

We call a vertex of the graph H *universal* if it is connected to every vertex of H. In particular, a universal vertex must be connected to itself too. The importance of the configuration graph is shown by the following theorem.

THEOREM 3.1. *Assume H has no universal vertex. Then G has a homomorphism to H obeying $\mathcal{P}$ if and only if there exists a directed path from S_0 to S_{z+1} in the configuration graph.*

Proof. Assume $\chi : V \to C$ is a homomorphism from G to H obeying $\mathcal{P}$. For $1 \leq i \leq z$ define the function B_i on C by setting $B_i(c)$ to be the largest integer $0 \leq B_i(c) \leq |L_i|$ satisfying that χ does not map any of the first $B_i(c)$ vertices of L_i to c. Clearly, B_i takes the value 0 on $\chi(x)$ for the first vertex x of L_i. We know that the vertices in the layer L_i have a common neighbor y in L_{i-1}. As $\chi(y)$ is not universal in H, there must exist $c \in C$ not adjacent to $\chi(y)$, and thus χ cannot take the value c on any neighbor of y making $B_i(c) = |L_i|$. Thus $S_i = (i, B_i)$ is a configuration. We claim that $S_0S_1 \ldots S_zS_{z+1}$ is a directed path in the configuration graph. Indeed, for $0 \leq i \leq z$ the restriction of χ to L_i provides for the edge $\overrightarrow{S_iS_{i+1}}$. Conditions 1 and 2 are satisfied trivially; to see condition 3, one has to use our observation that the neighbors in L_{i+1} of any vertex $x \in L_i$ are the first $d_+(x)$ vertices of that layer.

Conversely, let us assume that there is a directed path from S_0 to S_{z+1} in the configuration graph. By the layered structure of the configuration graph this path must be of the form $S_0S_1 \ldots S_zS_{z+1}$ with $S_i = (i, B_i)$ and appropriate functions B_i. For $0 \leq i \leq z$, let $\chi_i : L_i \to C$ be a homomorphism providing for the edge $\overrightarrow{S_iS_{i+1}}$ and let $\chi : V \to C$ be the union of these maps. We claim that χ is a G to H homomorphism obeying $\mathcal{P}$.

The function χ obeys $\mathcal{P}$ since all its parts χ_i do so by condition 1.

To see that χ is a homomorphism we have to show that the image of every edge $xy \in E$ is an edge in H. Clearly, x and y have to come from the same or neighboring layers. If they are in the same layer L_i, then $\chi(x)\chi(y) = \chi_i(x)\chi_i(y) \in E(H)$ because χ_i is a homomorphism. Now assume that for some $0 \leq i < z$ we have vertices $x \in L_i$ and $y \in L_{i+1}$ such that their images $\chi(x) = \chi_i(x)$ and $\chi(y) = \chi_{i+1}(y)$ are not adjacent in H. By condition 2, χ_{i+1} does not map the first $B_{i+1}(\chi(y))$ vertices of L_{i+1} to $\chi(y)$. Thus y is not among the first $B_{i+1}(\chi(y))$ vertices of L_{i+1}. By condition 3 on χ_i we have $B_{i+1}(\chi(y)) \geq d_+(x)$, so y is not among the first $d_+(x)$ vertices of L_{i+1}, so it is not adjacent to x as needed. ◻

Our next theorem tells us how to construct the configuration graph, more precisely, how to decide whether an edge is present. Let us fix two configurations $S = (i, B)$ and $S' = (i + 1, B')$. Let G_i be the subgraph of G induced on the layer L_i, and let us define a list mapping $\mathcal{P}'$ on G_i as follows. For $1 \leq j \leq |L_i|$, let x_j stand for the jth vertex in the layer L_i and let us set $\mathcal{P}'(x_j) = \{c \in \mathcal{P}(x_j) \mid B(c) < j$ and $\forall c' \in C$ either $d_+(x_j) \leq B'(c')$ or $cc' \in E(H)\}$.

THEOREM 3.2. *With S, S', G_i, and $\mathcal{P}'$ as above there is an edge from S to S' in the configuration graph if and only if G_i has a homomorphism to H obeying $\mathcal{P}'$.*

Proof. Any homomorphism providing for $\overrightarrow{SS'}$ obeys $\mathcal{P}'$ by conditions 1–3. Conversely, any homomorphism from G_i to H that obeys $\mathcal{P}'$ provides for this edge. ☐

We now present our algorithm for the list H-coloring problem for graphs with all connected induced subgraphs having a multichain ordering.

We are given a fixed graph H, an input graph G, and the list mapping $\mathcal{P}$. We start with very simple reductions.

If H has a universal vertex c, then consider the subgraph G' of G induced by the vertices whose lists do not contain c. Clearly, G has a homomorphism to H obeying $\mathcal{P}$ if and only if G' has such a homomorphism, as the vertices outside G' can "freely" be mapped to c.

Let H' stand for the subgraph of H induced by all the vertices that appear in the lists in $\mathcal{P}$. Clearly, G has a homomorphism to H obeying $\mathcal{P}$ if and only if G has a homomorphism to H' obeying $\mathcal{P}$.

G has a homomorphism to H obeying $\mathcal{P}$ if and only if all connected components of G have homomorphisms to H obeying $\mathcal{P}$.

We use these reductions (repeatedly if necessary) until we arrive at a problem in which G is connected, H has no universal vertex, and each vertex of H appears on a list of $\mathcal{P}$.

Start by constructing the layers $L_0, \ldots, L_z$ of a multichain ordering of G with the corresponding ordering of the vertices within the layers according to decreasing d_- degrees. Construct the configurations for this multichain ordering, including S_0 and S_{z+1}. Construct the edges of the configuration graph using a recursive call to check for the presence of each possible edge using the equivalent condition, as given in Theorem 3.2. Return TRUE if there is directed path from S_0 to S_{z+1} in the configuration graph, and return FALSE otherwise.

Note that the recursive calls to determine the presence of an edge from the configuration (i, B) to $(i+1, B')$ is simpler than the original problem instance. Indeed, it is a list H-coloring problem for G_i and G_i has a single vertex for $i = 0$, while for $i > 0$ we have a vertex c of H with $B(c) = |L_i|$ and this vertex does not show up in any of the lists—basically decreasing the number of vertices in the target graph H. To give base to this recursion we solve the trivial instances directly: If either G or H has a single vertex, deciding the list H-coloring problem for G becomes trivial. We can also handle the case where H has two vertices directly. If the two vertices are not adjacent in H, we must map each connected component of G to one or the other vertex. If the two vertices of H are connected and there is no loop in H, we face a 2-list coloring problem already discussed in the introduction. Finally if the two vertices of H are connected and there is also a loop in H, then H has a universal vertex and list H-coloring reduces to list H'-coloring with H' having a single vertex.

Using Theorems 3.1 and 3.2 it is straightforward to see that the algorithm below correctly answers the question of whether G has a homomorphism to H obeying $\mathcal{P}$.

It is a bit more involved to estimate the running time. Let k and n stand for the number of vertices in H and G. We claim that the running time of the algorithm is $O(n^{k^2-3k+4})$ (the constant of proportionality depends on k). We prove this statement by induction on k. For $k \leq 2$ the algorithm clearly finishes in time $O(n^2)$. Let us

Algorithm 1 LH(G, $\mathcal{P}$, H).

```
Input: Graphs $G$, $H$, list mapping $\mathcal{P}$ where every connected induced subgraph of
$G$ must have a multichain ordering
Output: TRUE if there is a homomorphism from $G$ to $H$ obeying $\mathcal{P}$; FALSE
otherwise
Let $H'$ be the subgraph of $H$ induced by vertices that appear in at least one list of
$\mathcal{P}$.
if $H' \neq H$ then return LH($G$, $\mathcal{P}$, $H'$)
end if
if $H$ has a universal vertex $c$ then
    Let $G'$ be the subgraph of $G$ induced by the vertices $x$ with $c \notin \mathcal{P}(x)$,
        and let $\mathcal{P}'$ be the restriction of $\mathcal{P}$ to this subgraph. return LH($G'$, $H$, $\mathcal{P}'$)
end if
if $G$ has a single vertex then
    if $H$ has a loop or $G$ has no loop and $H$ has at least one vertex then return
TRUE
    else return FALSE
    end if
end if
for each connected component $D = (V, E)$ of $G$ do
    if $H$ has at most two vertices then
        Find all (at most two) homomorphisms from $D$ to $H$.
        if at least one of the homomorphisms obeys $\mathcal{P}$ then
            $c_D \leftarrow$ TRUE
        else
            $c_D \leftarrow$ FALSE
        end if
    else
        Find a multichain ordering $L_0, \ldots, L_z$ of $D$, and order the vertices of
            each layer by decreasing size of neighborhood in the next layer.
        Initialize the directed configuration graph to have a vertex for each
            configuration of this multichain ordering, including $S_0$ and $S_{z+1}$.
        for $i \leftarrow 0$ to $z$ do
            Let $D_i$ be the subgraph of $D$ induced by $L_i$.
            for each pair of configurations $S = (i, B)$ and $S' = (i+1, B')$ do
                Construct a list mapping $\mathcal{P}'$ for $D_i$ as follows.
                for $j \leftarrow 1$ to $|L_i|$ do
                    Let $x_j$ stand for the $j$th vertex in the layer $L_i$.
                    $\mathcal{P}'(x_j) \leftarrow \{c \in \mathcal{P}(x_j) \mid B(c) < j$
                        $\forall c' \in V(H)(d_+(x_j) \leq B'(c')$ or $cc' \in E(H))\}$
                end for
                if LH($D_i$, $\mathcal{P}'$, $H$) = TRUE then
                    Add edge $\overrightarrow{SS'}$ to the configuration graph
                end if
            end for
        end for
```

Algorithm 1 LH(G, $\mathcal{P}$, H) (continued).

```
            if there is a directed path from $S_0$ to $S_{z+1}$ in the configuration graph then
                $c_D \leftarrow$ TRUE
            else
                $c_D \leftarrow$ FALSE
            end if
        end if
    end for
    if $c_D =$ TRUE for all components $D$ of $G$ then return TRUE
    else return FALSE
    end if
```

assume $k > 2$. If H has a universal vertex, our reduction reduces list H-coloring to a single list H'-coloring instance with H' having fewer vertices. If H has no universal vertex, we split G into connected components, find the multichain ordering of each component, and build the configuration graphs corresponding to them. The number of configurations for a fixed layer L_i of a single component is $O(|L_i|^{k-2})$ because the value of the function B in a configuration (i, B) is arbitrary for $k-2$ vertices of H, but it has to be either 0 or $|L_i|$ for two vertices of H. So the number of configurations for all connected components together can be bounded by $O(n^{k-2})$ and the number of potential edges (the number of recursive calls on the top level) is $O(n^{2k-4})$. In a recursive call to test the presence of an edge in the configuration graph one uses a list mapping that avoids at least one vertex of H completely, so the inductive hypothesis can be used for $k-1$. The only exception to this rule is the test for an edge leaving the configuration S_0 of one of the components, but there the recursive call is for a trivial graph on $|L_0| = 1$ vertices. These trivial recursive calls take constant time, and the other recursive calls take $O(n^{(k-1)^2-3(k-1)+4})$ time, so all recursive calls finish in $O(n^{2k-4}n^{(k-1)^2-3(k-1)+4}) = O(n^{k^2-3k+4})$ time. This huge time bound clearly dominates the time of the nonrecursive part of the algorithm.

4. Conclusion.

We have given a polynomial-time algorithm to solve the list H-coloring problem for fixed H if every connected induced subgraph of the input graph has a multichain ordering. Graphs satisfying this property form a subclass of weakly chordal graphs that contains all interval graphs and all permutation graphs, and even some graphs that have asteroidal triples.

REFERENCES

[1] M. Biro, M. Hujter, and Z. Tuza, *Precoloring extension.* I. *Interval graphs*, Discrete Math., 100 (1992), pp. 267–279.

[2] K. S. Booth and G. S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.

[3] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes: A Survey*, SIAM, Philadelphia, 1999.

[4] A. Brandstädt and V. V. Lozin, *On the linear structure and clique-width of bipartite permutation graphs*, Ars Combin., 67 (2003), pp. 273–281.

[5] J. Díaz, M. Serna, and D. M Thilikos, *Counting H-colorings of partial k-trees*, Theoret. Comput. Sci., 281 (2002), pp. 291–309.

[6] S. Gravier, D. Kobler, and W. Kubiak, *Complexity of list coloring problems with a fixed total number of colors*, Discrete Appl. Math., 117 (2002), pp. 65–79.

[7] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*, Oxford University Press, Oxford, UK, 2004.
[8] C. Hoàng, M. Kamiński, V. Lozin, J. Sawada, and X. Shu, *Deciding k-colorability of P_5-free graphs in polynomial time*, Algorithmica, 57 (2010), pp. 74–81.
[9] K. Jansen and P. Scheffler, *Generalized coloring for tree-like graphs*, Discrete Appl. Math., 75 (1997), pp. 135–155.
[10] T. R. Jensen and B. Toft, *Graph Coloring Problems*, John Wiley & Sons, New York, 1994.
[11] J. Kratochvíl and Z. Tuza, *Algorithmic complexity of list colorings*, Discrete Appl. Math., 50 (1994), pp. 297–302.
[12] R. M. McConnell and J. P. Spinrad, *Modular decomposition and transitive orientation*, Discrete Math., 201 (1999), pp. 189–241.
[13] A. Pnueli, A. Lempel, and S. Even, *Transitive orientation of graphs and identification of permutation graphs*, Canad. J. Math., 23 (1971), pp. 160–175.
[14] J. A. Telle and A. Proskurowski, *Algorithms for vertex partitioning problems on partial k-trees*, SIAM J. Discrete Math., 10 (1997), pp. 529–550.
[15] P. Valadkhan, *List Matrix Partitions of Special Graphs*, Ph.D. thesis, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, 2013.
[16] M. Yannakakis, *The complexity of the partial order dimension problem*, SIAM J. Algebraic Discrete Methods, 3 (1982), pp. 351–358.