Journal of Combinatorial Theory, Series B **72**, 236–250 (1998)
Article No. TB971812

List Homomorphisms to Reflexive Graphs

Tomas Feder

IBM Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099

and

Pavol Hell

School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6

Received February 6, 1996

Let H be a fixed graph. We introduce the following list homomorphism problem: Given an input graph G and for each vertex v of G a "list" $L(v) \subseteq V(H)$, decide whether or not there is a homomorphism $f: G \to H$ such that $f(v) \in L(v)$ for each $v \in V(G)$. We discuss this problem primarily in the context of reflexive graphs, i.e., graphs in which each vertex has a loop. We give a polynomial time algorithm to solve the problem when H is an interval graph and prove that when H is not an interval graph the problem is NP-complete. If the lists are restricted to induce connected subgraphs of H, we give a polynomial time algorithm when H is a chordal graph and prove that when H is not chordal the problem is again NP-complete. We also argue that the complexity of certain other modifications of the problem (including the retract problem) are likely to be difficult to classify. Finally, we mention some newer results on irreflexive and general graphs.

1. INTRODUCTION

We consider finite undirected graphs which may have loops. A graph is *reflexive* if each vertex has a loop, and *irreflexive* if no vertex has a loop. A *homomorphism* $f: G \to H$ of a graph G to a graph H is a vertex mapping which preserves edges, i.e., a mapping $f: V(G) \to V(H)$ such that $uv \in E(G)$ implies $f(u)\, f(v) \in E(H)$. If there is a homomorphism of G to H we write $G \to H$. Note that a homomorphism may in general identify adjacent vertices (onto a vertex which has a loop). Loops are usually ignored when properties of reflexive graphs are discussed. In particular, we say that a reflexive graph H is a *tree*, *cycle*, *interval graph*, or *chordal graph*, precisely when the graph obtained from H by removing all loops has the corresponding property (as defined, e.g., in [14]). As the only exception to this,

0095-8956/98 $25.00

we apply the definition of a bipartite graph literally, and hence, a bipartite graph is by definition irreflexive.

Let H be a fixed graph; we shall refer to the problem of deciding whether or not an input graph G satisfies $G \to H$ as the *homomorphism problem* HOMH.

The problem HOMH is only interesting for irreflexive graphs, since if H has a loop on a vertex v then every G admits a homomorphism to H, by mapping all vertices of G to v. When H is the irreflexive complete graph K_n then a homomorphism of G to H is just an n-colouring of G. Thus HOMK_n is the problem of n-colourability, which is known to be NP-complete when $n > 2$, and polynomial time solvable for $n = 1, 2$. The complexity of HOMH for other irreflexive graphs has been classified in [18]: When H is any bipartite graph then $G \to H$ if and only if G is also bipartite; thus in this case HOMH is polynomial. When H is not bipartite then HOMH is NP-complete [18].

The problem HOMH can also be stated for a digraph H. (Digraph homomorphisms must also preserve the directions of edges.) This problem seems much harder [5, 6] and there is some evidence [8] that a full classification of the complexity of HOMH for digraphs H may be difficult.

Before formally introducing list homomorphisms, we wish to point out an analogy with list colourings, which have been studied in graph theory for some time [7, 2, 21, 24, 26]. Informally, list colourings restrict at each vertex the set of allowed colours. The study of list colourings introduced deep new insights and techniques which led to solutions of old problems on ordinary colourings, as well as to fruitful new conjectures [2, 7, 12]. In some cases, the list equivalents of old problems turned out to be in a sense nicer than the original problems. Consider, for instance, planar graphs. Every planar graph is 4-colourable, and there is a planar graph which is not 3-colourable. The former statement is extremely difficult to prove [1], while the second one is trivial (take K_4). The situation is more balanced when one asks the corresponding question for list colourings: Specifically, Erdős, Rubin, and Taylor [7] define a graph G to be *k-choosable* if a proper colouring of G exists whenever each vertex of G has a list of k allowed colours to choose from. It turns out that every planar graph is 5-choosable and that there is a planar graph which is not 4-choosable. Both of these results are nontrivial and have very nice proofs of manageable size [26, 24].

Suppose H is a fixed graph. Given a graph G and lists $L(v) \subseteq V(H)$ for all $v \in V(G)$, a *list homomorphism* of G to H, with respect to the lists L, is a homomorphism $f: G \to H$ with $f(v) \in L(v)$ for each $v \in V(G)$. Thus list colourings are precisely list homomorphisms to irreflexive complete graphs. The *list homomorphism problem* L-HOMH asks, for input G with lists L, whether there is a list homomorphism of G to H with respect to L. Since list homomorphisms limit the possible images of each vertex, list homomorphism problems may be interesting even when H has loops. In fact we shall concentrate in this paper on the case of reflexive graphs. For irreflexive and general graphs we have similar results, summarized in the last section; we will return to this topic in separate papers [10, 11].

The list homomorphism problem for reflexive graphs admits a very natural interpretation. Suppose we have a set of processors, with direct connections between certain pairs of processors. Define a graph H with the processors as vertices and the processor connections as edges. Suppose we also have a set of jobs to be performed by the processors, each job being suitable only for some of the processors, and with certain pairs of jobs requiring communication during processing. We may now define another graph G with the jobs as vertices and the job pairs in need of communication as edges; in addition, we also have for each job a list of admissible processors. Then a desired assignment of jobs to processors is precisely a list homomorphism of G to H. Thus we have an instance of the problem L-HOMH. (A similar interpretation would apply for emulating one architecture by another.) An important point here is that each processor is naturally viewed as being able to communicate with itself; thus the graphs in question have to be reflexive.

We also consider two variants of the list homomorphism problem. The *connected list homomorphism problem* CL-HOMH restricts L-HOMH to those inputs in which each list $L(v)$ induces a connected subgraph of H. The *one-or-all list homomorphism problem* OAL-HOMH restricts the inputs of L-HOMH to have each list $L(v)$ either a single vertex of H or the entire set $V(H)$. When H itself is connected, OAL-HOMH is a further restriction of CL-HOMH. The problem OAL-HOMH is equivalent to a previously studied problem of *graph retraction*.

For reflexive graphs we have the following results: In Section 2 we prove that there is a polynomial time algorithm for L-HOMH when H is an interval graph, and L-HOMH is NP-complete otherwise. In Section 3 we show that there is a polynomial time algorithm for CL-HOMH when H is a chordal graph, and CL-HOMH is NP-complete otherwise. In contrast to these results we shall argue in Section 4 that the boundary between easy and hard OAL-HOMH problems is unlikely to be simple to describe. Finally, in Section 5 we mention some recent results on list homomorphisms for irreflexive and general graphs.

In analogy to the above discussion on list colourings of planar graphs, one can argue that the list homomorphism versions of the complexity questions are in a sense nicer than the original homomorphism questions: Both in the reflexive case (the present paper) and the irreflexive case ([10]) we obtain both interesting new polynomial algorithms for list homomorphisms to well-structured graphs, and natural NP-completeness constructions for list homomorphisms to unstructured graphs. In contrast, for ordinary homomorphism problems, the reflexive case is trivial, and in the irreflexive case, it is quite difficult to prove that HOMH is NP-complete for *all* nonbipartite H [18], while the polynomial algorithms for HOMH with H bipartite are trivial (cf. above).

2. LIST HOMOMORPHISMS WITH GENERAL LISTS

In this section we shall assume that all graphs are reflexive, unless stated otherwise. A *chordless cycle* in a graph H is an induced cycle, of length at least four, without chords. We prove the following fact:

THEOREM 2.1. *If H contains a chordless cycle then* L-HOMH *is NP-complete.*

For future reference we first prove a stronger result for a special case. This result has been independently proved by G. MacGillivray (personal communication).

LEMMA 2.2. *If H is the cycle of length k, with $k \geqslant 4$, then* OAL-HOMH *is NP-complete.*

Proof. It is clear that OAL-HOMH is in NP. We present a reduction from k-colourability of irreflexive graphs: For any irreflexive graph X we construct (in polynomial time) a graph G with lists $L(g) \subseteq V(H)$, $g \in V(G)$, in such a way that X is k-colourable if and only if G admits a list homomorphism to H. First of all, the graph G will contain a fixed copy H_0 of H. Next, each edge xy of X will be replaced by a gadget. If k is even, $k = 2l$ ($l \geqslant 2$), then the gadget consists of a new vertex, z, a path of length $l-1$ between x and z, and two paths of length l between z and y. If k is odd, $k = 2l+1$ ($l \geqslant 2$), then the gadget consists of three new vertices z, z', z'' forming a triangle, a path of length $l-1$ between x and z, and two paths of length l between z' and y, and between z'' and y. (All the paths are internally disjoint; cf. Fig. 1.) Note that each edge xy of X gives rise to its own copy H_{xy} of H (formed by the two paths joining z, y in case k is even, or

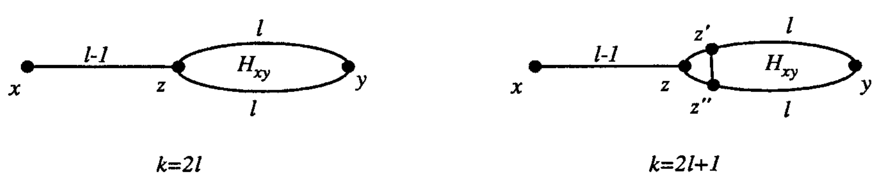

Fig. 1. The gadgets for the proof of the lemma.

by the edge $z'z''$ and the two paths joining z', y and z'', y in case k is odd). Finally, each H_{xy} is connected to H_0 by a sequence of $k-1$ new copies of H, where consecutive copies are joined so that the ith vertex of one copy is joined to the ith and the $(i+1)$th vertex of the next copy (with addition modulo k; see Fig. 2). The lists are defined as follows: for each vertex $g \in V(H_0)$ we let $L(g) = \{g\}$; for all other vertices $g \in V(G)$ we let $L(g) = V(H)$. Note that the connections between consecutive copies of H in each chain connecting an H_{xy} to H_0 assure that in any list homomorphism of G to H the ith vertex of one copy can be only identified with the ith or $(i+1)$th vertex of the next copy. Thus each cycle must "rotate" to the next cycle. It is not hard to conclude (using the fact that $l \geqslant 2$) that in any list homomorphism of G to H, the vertices y and z of the gadget corresponding to the edge xy (this is the gadget containing the copy H_{xy} of the cycle H) must map to opposite vertices of H. Here opposite means of distance $l = \lfloor k/2 \rfloor$ along the cycle, regardless of the parity of k. Moreover, for any two opposite vertices u, v of H_0, there is a list homomorphism of G to H which maps x to u and y to v. Thus each list homomorphism of G to H takes x, y to different vertices of H, and any two different vertices of H are the images of x, y under some list homomorphism of G to H. Therefore, G admits a list homomorphism to H if and only if the vertices of X can be labelled by the vertices of H so that adjacent vertices of X obtain different labels, i.e., if and only if X is k-colourable. ∎

Note that when H is the reflexive three-cycle then OAL-HOMH is polynomial time solvable; indeed this is so for any reflexive complete graph.

Now we can easily complete **the proof of the theorem**: It is again clear that L-HOMH is in NP. Let H' be an induced cycle (of length $k \geqslant 4$) without chords in H. Any instance of OAL-HOMH' can be viewed as an instance of L-HOMH since H' is a subgraph of H. Thus OAL-HOMH' is a restriction of L-HOMH and hence the lemma implies that the latter is also NP-complete.

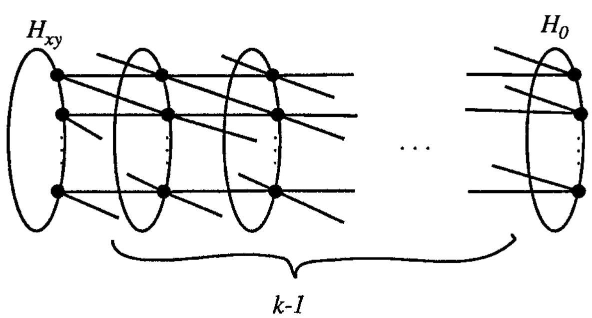

Fig. 2. Connections between the copies of H.

An *asteroidal triple* of H is a set of three nonadjacent vertices $0, 1, 2 \in V(H)$ such that for each pair $i \neq j$ of vertices from $\{0, 1, 2\}$ there is a path in H joining i and j not containing any neighbour of the third vertex k ($k \neq i, j$) from $\{0, 1, 2\}$.

THEOREM 2.3. *If H contains an asteroidal triple then* L-HOMH *is* NP*-complete.*

Proof. Let 0, 1, 2 be three nonadjacent vertices of H and let, for each pair of distinct $i, j \in \{0, 1, 2\}$, $P_{i,j}$ be a shortest path in H joining i and j and not containing any neighbours of the third vertex $k \in \{0, 1, 2\}$. Note that the assumption that $P_{i,j}$ is a shortest path implies that i (and j) has only one neighbour on $P_{i,j}$.

We reduce not-all-equal 3-satisfiability without negated variables [13] to L-HOMH in a manner similar to the proofs in [20]. The key in this reduction is the construction of "choosers." Let i, j be distinct elements of $\{0, 1, 2\}$ and let I, J be distinct subsets of $\{0, 1, 2\}$. An (i, I, j, J)-chooser is a path P with endpoints a and b, and with lists $L_p \subseteq V(H)$, $p \in V(P)$, such that any list homomorphism f of P to H has $f(a) = i$ and $f(b) \in I$ or $f(a) = j$ and $f(b) \in J$, and, moreover, for any $i' \in I$ and any $j' \in J$ there exists a list homomorphism $f\colon P \to H$ with $f(a) = i$ and $f(b) = i'$ and a list homomorphism $g\colon P \to H$ with $g(a) = j$ and $g(b) = j'$.

Suppose that P is a $(0, \{0, 1\}, 1, \{1, 2\})$-chooser, P' is a $(0, \{1, 2\}, 1, \{2, 0\})$-chooser, and P'' is a $(0, \{2, 0\}, 1, \{0, 1\})$-chooser. Let T be the tree obtained from P, P', P'' by identifying all three vertices b, calling the three a vertices a, a', a''.

It is easy to check that T admits a list homomorphism to H in which the images of a, a', a'' are any combination of 0 and 1 except 000 and 111. Hence if we replace, in an instance of not-all-equal 3-satisfiability without negated variables, each clause by a copy of T (with the literals identified

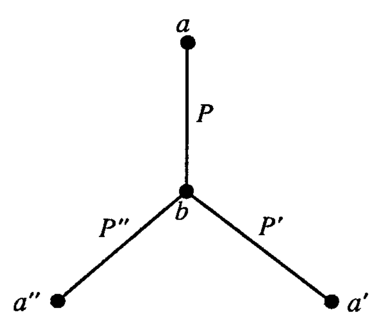

Fig. 3. The tree T constructed from three choosers.

with a, a', a''), we obtain a graph G which admits a list homomorphism to H if and only if the instance is satisfiable.

Thus it remains to construct the required (i, I, j, J)-choosers. Suppose first that $I=\{i, k\}$ and $J=\{j, k\}$, $k \neq i, j$. Let l be the larger of the lengths of $P_{i,k}$, $P_{j,k}$, and let A be the set of all vertices in all the paths $P_{i,j}$, $P_{i,k}$, $P_{j,k}$, except those which are adjacent to i or j. (Recall that there are at most two vertices adjacent to i and at most two adjacent to j on these paths.) Let i^+, j^+ be the first vertices on $P_{i,k}$, $P_{j,k}$ after i, j, respectively. Let P be a path of length l from a to b, and let a^+ denote again the first vertex after a on P. We define $L(a)=\{i, j\}$, $L(a^+)=\{i^+, j^+\}$, $L(b)=\{0, 1, 2\}$, and $L(p)=A$ for all other $p \in P$. Then P with L is an (i, I, j, J)-chooser:

Indeed, any list homomorphism of P to H which takes a to i must take a^+ to i^+ as i, j^+ are not adjacent (since j^+ lies on the path $P_{j,k}$). Then the next vertex on P is taken either to i, in which case all subsequent vertices including b will map to i, or to the next vertex on $P_{i,k}$, in which case b cannot map to either i or j because all their neighbours are absent from the label set A. Since the length of P is at least as large as that of $P_{i,j}$, there is a list homomorphism of P to H taking a to i and b to k; there is also an obvious list homomorphism taking both a and b (and all other vertices of P, except a^+) to i. List homomorphisms taking a to j are analyzed in an analogous fashion.

We next consider the case when $I=\{i\}$ and $J=\{k\}$, $k \neq j$. Let l be the length of the path $P_{j,k}$ and let $A=V(P_{j,k})$. Let P be a path of length l from a to b, with all lists $L(p)$ equal to A except $L(a)=\{i, j\}$ and $L(b)=\{i, k\}$. Then a simple analysis shows that P with L is an (i, I, j, J)-chooser.

It is now possible to form all the required choosers by concatenating the two kinds of choosers constructed above:

The required $(0, \{0, 1\}, 1, \{1, 2\})$-chooser P is obtained by concatenating a $(0, \{0\}, 1, \{2\})$-chooser and a $(0, \{0, 1\}, 2, \{2, 1\})$-chooser. The required $(0, \{1, 2\}, 1, \{2, 0\})$-chooser P' is obtained by concatenating a $(0, \{0\}, 1, \{2\})$-chooser, a $(0, \{1\}, 2, \{2\})$-chooser, a $(1, \{1\}, 2, \{0\})$-chooser, and a $(1, \{1, 2\}, 0, \{0, 2\})$-chooser. Finally, the required $(0, \{2, 0\}, 1, \{0, 1\})$-chooser P'' is obtained by concatenating a $(0, \{2\}, 1, \{1\})$-chooser and a $(2, \{0, 2\}, 1, \{0, 1\})$-chooser. ∎

COROLLARY 2.4. *If H is not an interval graph then* L-HOMH *is NP-complete.*

Proof. If H is not an interval graph then it contains a chordless cycle or an asteroidal triple [14]. ∎

THEOREM 2.5. *If H is an interval graph then* L-HOMH *is polynomial time solvable.*

Proof. We shall give a polynomial time reduction of this problem to the (polynomial time solvable) problem of 2-satisfiability [13]. Thus suppose H is an interval graph and G with $L(v), v \in V(G)$, is an instance of L-HOMH. We may assume that the vertices of H, and hence the lists $L(v)$, are given as intervals (with two intervals intersecting if and only if the corresponding vertices are adjacent in H). In addition, we may assume [14] that the endpoints of the intervals are distinct and order them as $a_1 < a_2 < \cdots < a_{2k}$, where $k = |V(H)|$. Let P denote a set of $2k+1$ new points $p_0, p_1, ..., p_{2k}$ such that $p_0 < a_1$, $p_{2k} > a_{2k}$, and $a_i < p_i < a_{i+1}$ for $i = 1, 2, ..., 2k-1$. The corresponding instance of 2-satisfiability will have variables $l_{v,p}$ and $r_{v,p}$ for each vertex v of G and each point $p \in P$. We shall define the clauses in such a way that a list homomorphism of G to H, with respect to the lists L, exists if and only if the clauses can be satisfied. The intented meaning of $l_{v,p} = 1$ is "the image of v is an interval with the left endpoint less than p;" and $r_{v,p} = 1$ is intended to mean "the image of v is an interval with the right endpoint greater than p." (Once the values $l_{v,p}$ and $r_{v,p}$, for all $p \in P$, are decided, we will be able to choose an actual interval as the image of v). To assure that adjacent vertices are assigned intersecting intervals we state a clause $l_{u,p} \vee r_{v,p}$, for each $p \in P$ and each *ordered* pair u, v of adjacent vertices of G. Since G is a reflexive graph, this will include the clauses $l_{u,p} \vee r_{u,p}$, which assure that each interval has its left endpoint less than its right endpoint. To reflect the lists $L(v)$ we impose clauses requiring that the interval chosen for each vertex v can be extended to an interval in $L(v)$. These are the clauses $\bar{l}_{v,p} \vee \bar{r}_{v,q}$ for each $v \in V(G)$ and each $p, q \in P$ such that the interval (p, q) is not contained in any interval from $L(v)$. Finally, we assure that at least one $l_{v,p}$ and at least one $r_{v,p}$ is true for each $v \in V(G)$ by stating the clauses $l_{v,p_{2k}}$ and r_{v,p_o} for each vertex v of G. We now claim that these clauses are satisfiable if and only if a list homomorphism exists. Indeed, if f is such a list homomorphism, we only need to set the values $l_{v,p}$ and $r_{v,p}$ as suggested above; i.e., $l_{v,p} = 1$ if and only if the left endpoint of the interval $f(v)$ is less than p, and $r_{v,p} = 1$ if and only if the right endpoint of the interval $f(v)$ is greater than p. On the other hand, a satisfying truth assignment can be used to define a homomorphism as follows: Let $p \in P$ be the smallest point with $l_{v,p} = 1$ and let $q \in P$ be the largest point with $r_{v,q} = 1$. Then there exists an interval $I \in L(v)$ which contains (p, q), and we define $f(v) = I$. The clauses assure that adjacent vertices are assigned intersecting intervals. Since H, and hence P, is fixed, this is a polynomial time reduction. ∎

We are grateful to X. Zhu for pointing out that an alternate proof of the above result can be obtained by proving that interval graphs have the X-underbar property defined in [15] and by modifying the algorithm in [15] to apply to the list homomorphism problem. The proof given here actually shows that the problem is of *strict width two*, in the terminology of [8].

3. CONNECTED LISTS

We again assume that all graphs in this section are reflexive. We say that a set of vertices of H is *connected* if it induces a connected subgraph of H. Thus CL-HOMH is the problem L-HOMH restricted to inputs with connected lists. It is easy to see that Theorem 2.1 applies in fact to CL-HOMH (all lists mentioned in the proof are connected).

THEOREM 3.1. *If H is not chordal then* CL-HOMH *is NP-complete.*

THEOREM 3.2. *If H is chordal then* CL-HOMH *is polynomial time solvable.*

Proof. We present a polynomial algorithm for CL-HOMH. First we find (in polynomial time [14]) a perfect elimination ordering $h_1, h_2, ..., h_n$ of H. (This means that for each $i = 1, 2, ..., n$, any two neighbours of h_i in $\{h_{i+1}, h_{i+2}, ..., h_n\}$ are adjacent.) Let G be a graph and suppose that, for each $g \in V(G)$, the set $L(g) \subseteq V(H)$ is connected. In the ith stage of the algorithm ($i = 1, 2, ..., n$), we will consider removing h_i from each list $L(g)$. If $L(g)$ contains h_i and at least one other vertex, we do remove h_i from $L(g)$. If $L(g)$ does not contain h_i we do nothing. Finally, if $L(g) = \{h_i\}$ then we do not change $L(g)$ but, for each g' adjacent to g, we remove from $L(g')$ all vertices which are not adjacent to h_i in H and we remove g from consideration. If at any time an empty list is produced then we declare that there is no list homomorphism; otherwise we terminate with singleton lists which define a desired list homomorphism.

Since the graph H is fixed, it is clear that this is a polynomial time algorithm. (In fact, with $|V(G)|$ processors, it can be performed in constant time.) We now prove that it is correct. Specifically, we shall show that if the original connected lists L admit a list homomorphism, then throughout the execution of the algorithm the changing lists L' are also connected and still admit a list homomorphism. (This is the case for the initial lists, by assumption.) This statement assures that at termination all lists are singletons and they define a list homomorphism. Thus assume the invariant holds at the outset of the ith stage; we claim that it remains true after the ith stage. In fact, these properties remain true after each list $L(g)$ is treated: If h_i is removed from $L(g)$ then the only list that changes is $L(g)$ and it remains connected, as any two neighbours of h_i are adjacent. Moreover, if a list homomorphism existed before $L(g)$ was treated and if it assigned h_i to g, then we can replace h_i by any h_j adjacent to h_i in H and still have a homomorphism. (Note that some h_j adjacent to h_i must belong to the current list of g, since the list is connected.) Indeed, suppose that f is a list homomorphism (with respect to the lists at the beginning of the ith stage) of G to H, with $f(g) = h_i$, and let f^* be defined by $f^*(x) = f(x)$ if $x \neq g$, and $f^*(g) = h_j$. To verify that f^* is a homomorphism, consider a vertex g' adjacent to g and suppose that $f(g') = f^*(g') = h_k$. If $k < i$ then h_k is the only element in the current list of g' and, hence, it is adjacent to all elements of the current list of g, including h_j. Thus suppose that $k > i$, and note that the fact that $f(g) = h_i$, $f(g') = h_k$ implies that h_i is adjacent to h_k in H. Since both j and k are greater than i, and both h_j and h_k are adjacent to h_i, we conclude (from the definition of a perfect elimination ordering) that $f^*(g) = h_j$ and $f^*(g') = h_k$ are adjacent in H as well. If $L(g) = \{h_i\}$ then the lists that change are $L(g')$ for g' adjacent to g in G. The change amounts to taking an intersection of the connected $L(g')$ with the neighbourhood of h_i, which is complete. Thus the intersection is connected (in fact, it is complete). Moreover, any list homomorphism with respect to the lists at the beginning of the ith stage remains a list homomorphism with respect to the lists at the end of the ith stage. ▮

COROLLARY 3.3. *If H is chordal then* OAL*H is polynomial time solvable.*

Proof. For connected graphs H, the problem OALH is a restriction of the problem CL-HOMH. The extension to disconnected H is straightforward, in view of the fact that a component of G cannot admit a list homomorphism to H if two of its vertices are required to map to different components of H. ▮

The essential feature of chordal graphs, on which the previous algorithm is based and which is easy to prove by contradiction, is the following.

THEOREM 3.4. *H is chordal if and only if for any connected sets A, $B \subseteq V(H)$, the intersection $A \cap N(B)$ is also a connected vertex set in H.*

(The neighbourhood $N(B)$ of the set B consists of all vertices adjacent to a vertex of B; since H is reflexive, it includes all vertices of B.)

We are grateful to N. Vikas for noticing the following alternate view of the above algorithm (cf. [25]), using the language of [20]: We first apply the edge consistency test. If it results in nonempty lists, then these lists are connected by the above theorem. It is easy to see that this fact allows one to define a homomorphism by choosing for each vertex the maximum element, in the perfect elimination order, of its list. In particular, this implies that CL-HOML is of *width one*, in the terminology of [8].

4. OAL-HOM*H* AND THE RETRACT PROBLEM RET*H*

The problem OAL-HOMH has been previously considered under an equivalent formulation. Suppose H is a subgraph of G. We say that H is a retract of G if there exists a homomorphism $f: G \to H$ (called *retraction*) such that $f(v) = v$ for all $v \in V(H)$. The retraction problem RETH for a fixed graph H takes as instance a graph G with a particular induced subgraph H' isomorphic to H and asks whether or not H' is a retract of G. Retraction problems in graph theory have been investigated since early seventies [16, 17, 22, 23, 4], often from an algorithmic perspective. The problem is meaningful for general graphs, although it has been studied mostly for reflexive [22] or irreflexive [16, 17] graphs. (Only [4] briefly discusses the general case.) For irreflexive graphs, the result of [18] implies that RETH is NP-complete when H is not bipartite. On the other hand, for bipartite H the problem RETH can be restricted to bipartite instances G—as there can be no retraction from a nonbipartite graph G to a bipartite graph H. Thus we discuss here two main versions of the retract problem RETH—one for reflexive graphs and one for bipartite graphs. We shall return to the problem RETH for general graphs H in the companion paper [11].

We first explain how the retract problem RETH corresponds to OAL-HOMH. Suppose we have an instance of the retract problem, i.e., an input graph G with a subgraph H' isomorphic to H. Then we can view G as an instance of OAL-HOMH by making all lists of vertices $v \in V(H')$ equal to $\{v\}$ and making the lists of all other vertices equal to $V(H')$. Clearly H' is a retract of G if and only if a list homomorphism exists. Conversely, suppose we have an instance of OAL-HOMH, i.e., a graph G with lists $L(v)$, $v \in V(G)$, which are either singletons or the entire $V(H)$. First, we may suppose that at most one vertex v of G has the list $\{h\}$ for any $h \in V(H)$. Otherwise we can identify in G all vertices with the list $\{h\}$. There will be a list homomorphism from the modified graph if and only if there was a list homomorphism from the old graph G. Second, we may assume that if v has the list $\{h\}$ and v' has the list $\{h'\}$, then vv' is an edge of G if and only if hh' is an edge of H. Indeed, if vv' is an edge and hh' is not, then there can be no list homomorphism; and if hh' is an edge of H then G admits a list homomorphism to H if and only if G with the edge vv' added, if it wasn't present, admits such a list homomorphism. It is clear that by adding extra vertices to G if necessary, we may assume that H is an induced subgraph of G, and a list homomorphism exists if and only if H is a retract of G. We have just proved:

THEOREM 4.1. *For every graph H, the problems* OAL-HOMH *and* RETH *are polynomially equivalent.*

In [16], it is proved that an irreflexive tree H is a retract of a bipartite graph G if and only if it is an isometric subgraph of G—that is, the distance between any two vertices of H is the same in G and in H. (More general results of this kind are proved in [3]; in particular the statement remains true for any chordal bipartite graph H.) A similar result has been proved for reflexive trees in [22]. These results yield polynomial time algorithms for RETH (and thus OAL-HOMH) when H is a reflexive or irreflexive tree. (In fact the above result of [3] implies a polynomial algorithm when H is a chordal bipartite graph, and

Corollary 3.3 implies a polynomial algorithm when H is a reflexive chordal graph.)

In [4], the authors observe a very close relationship between the bipartite and reflexive retract problems. We begin by making this relationship even more specific:

THEOREM 4.2. *For every bipartite graph H there exists a reflexive graph H′ such that* RETH *and* RETH' *are polynomially equivalent.*

Proof. Suppose H has a bipartition $V(H) = X \cup Y$. The graph H' is obtained from H by adjoining five new vertices n, n', s, s', and u such that n and n' are adjacent to each other and to all vertices in X, and s, s' are adjacent to each other and to all vertices in Y; the vertex u is adjacent to all vertices of H and to n' and s' (see Fig. 4). Moreover, all vertices of H' have loops (not depicted in the figures).

We first construct a polynomial time reduction from RETH to RETH'. Consider an arbitrary instance of RETH, i.e., a bipartite graph G containing H as a subgraph. Assume that $V(G) = X_G \cup Y_G$ is a bipartition of G such that $X \subseteq X_G$ and $Y \subseteq Y_G$. We construct a reflexive graph G' containing H' by adjoining to G the same five vertices n, n', s, s', u and making them adjacent to the appropriate vertices of H' as defined above, with the additional connections between n, n' and all vertices of $X_G - X$ and between s, s' and all vertices of $Y_G - Y$. We claim that H is a retract of G' if and only if H' is a retract of G'. Indeed, any retraction of G to H has a natural extension to a retraction of G' to H'; moreover, in any retraction of G' to H' those vertices of G that are not isolated in G lie on paths of length three joining n and s in G'—thus they must map to H. The isolated vertices in G may be mapped arbitrarily.

Now we construct a polynomial time reduction from RETH' to RETH. Thus let G' be a reflexive graph containing H'. We may assume that no vertex of $G' - H'$ is adjacent to both n and s, as otherwise H' is not a retract of G'. A similar argument shows that no vertex of $G' - H'$ can be adjacent to n, s' or

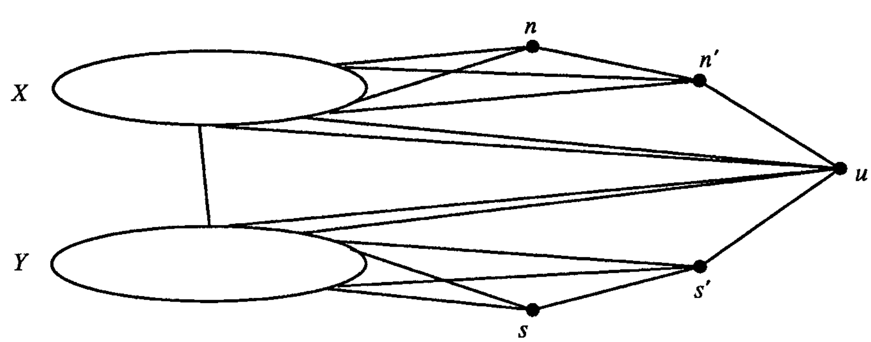

Fig. 4. The construction of H'.

n', s. Let A denote the set of vertices of $G' - H'$ adjacent to n which have distance two (in G') to s, and let A' denote the remaining set of vertices of $G' - H'$ adjacent to n. Similarly, let B denote the set of vertices of $G' - H'$ adjacent to s which have distance two (in G') to n, and let B' denote the remaining set of vertices of $G' - H'$ adjacent to s. Finally, let C denote the set of all other vertices of $G' - H'$. We denote by G^- the graph obtained from G' by removing the sets A', B', C. We claim that H' is a retract of G^- if and only if it is a retract of G'. Clearly, if H' is a retract of G' then it is also a retract of its subgraph G^-. Thus suppose r^- is a retraction of G^- to H'. Note that all vertices of $G^- - H'$ lie on paths of length three joining n and s, and therefore must be mapped to $X \cup Y$ by r^-. It follows from the definitions of A, A', B, B', C that there is no edge joining A' and $B \cup B' \cup Y \cup \{s, s'\}$, or B' and $A \cup A' \cup X \cup \{n, n'\}$, or C and $\{n, s\}$. Therefore we may define a retraction $r'\colon G' \to H'$ as follows:

$$
\begin{aligned}
r'(x) &= r^-(x) && \text{for all vertices of } G^- \\
r'(x) &= n' && \text{for all vertices of } A' \\
r'(x) &= s' && \text{for all vertices of } B' \\
r'(x) &= u && \text{for all other vertices of } C.
\end{aligned}
$$

We now consider possible retractions of G^- to H': the vertices of A will have to map to X and the vertices of B will have to map to Y. Let X_G denote the set of vertices obtained from $X \cup A$ by identifying any adjacent vertices. (The images of such vertices under any retraction of G^- to H' will have to be the same.) Let Y_G be obtained in a similar fashion from $Y \cup B$. Now we define a bipartite graph G to consist of X_G and Y_G (with all loops removed). It is easy to see that H is a retract of G if and only if H' is a retract of G^- and hence of G'. ∎

The preceding result will be useful to limit our attempts to classify the complexity of RETH. According to [8] for every constraint satisfaction problem Π there exists a bipartite graph H such that RETH and Π are polynomially equivalent. Since for constraint satisfaction problems it is not even known whether or not each Π is polynomial time solvable or NP-complete [8], this is taken as evidence that the complexity of bipartite retraction problems is not likely to be easily classified. (A similar result holds for the digraph homomorphism problem mentioned in the introduction [8].) In the same spirit, we interpret the next corollary to mean that the complexity of reflexive retract problems is a difficult question.

COROLLARY 4.3. *For every constraint satisfaction problem Π there exists a reflexive graph H such that* RETH *and Π are polynomially equivalent.*

Thus completely classifying the complexity of RETH for reflexive graphs H also seems difficult. Our results in this paper tell us that RETH is NP-complete when H is a cycle of length at least four and is polynomial time solvable when H is chordal. Nonchordal graphs H with polynomial-time solvable RETH can be constructed, for instance, by taking strong products of paths (cf. [19]).

5. CONCLUSIONS

We have shown that nicely structured graphs H can lead to interesting list homomorphism problems and that we can often determine precisely the boundary between easy and hard list homomorphism problems. This is somewhat surprising, as it seems hopeless to classify the complexity of some very similar homomorphism problems.

In a companion paper, joint with J. Huang [10], we study the list homomorphism problem for irreflexive graphs. We give a full classification of the complexity of L-HOMH when H is an irreflexive graph. As mentioned before, this problem is NP-complete when H is not bipartite. It turns out that among bipartite graphs the problem is still NP-complete when the complement of H is not a circular arc graph. When H is a bipartite complement of a circular arc graph, we shall give a polynomial time algorithm.

We also have results on general graphs (graphs in which each vertex may or may not have a loop) [11]. Let V_L denote the set of vertices of H which have loops. We shall prove that if H is connected but V_L is not, then RETH (and hence CL-HOMH and L-HOMH) are NP-complete. Restricting ourselves to general *trees* H, we prove that if V_L is connected, then CL-HOMH (and hence also RETH) are polynomial time solvable. Finally, we shall give a complete classification of general trees H into those with an NP-complete L-HOMH and those with a polynomial time solvable L-HOMH.

We are grateful to J. Huang for many valuable suggestions that improved this manuscript.

REFERENCES

1. K. Appel, W. Haken, and J. Koch, Every planar map is four colorable, *Illinois J. Math.* **21** (1977), 429–567.
2. N. Alon and M. Tarsi, Colourings and orientations of graphs, *Combinatorica* **12** (1992), 125–134.
3. H. J. Bandelt, A. Dahlmann, and H. Schutte, Absolute retracts of bipartite graphs, *Discrete Appl. Math.* **16** (1987), 191–215.

4. H. J. Bandelt, M. Farber, and P. Hell, Absolute reflexive retracts and absolute bipartite retracts, *Discrete Appl. Math.* **44** (1993), 9–20.
5. J. Bang-Jensen and P. Hell, On the effect of two cycles on the complexity of colouring, *Discrete Appl. Math.* **26** (1990), 1–23.
6. J. Bang-Jensen, G. MacGillivray, and P. Hell, The complexity of colouring by semicomplete digraphs, *SIAM J. on Discrete Math.* **1** (1988), 281–298.
7. P. Erdős, A. L. Rubin, and H. Taylor, Choosability in graphs, *Congr. Numer.* **26** (1979), 125–157.
8. T. Feder and M. Y. Vardi, Monotonic monadic SNP and constraint satisfaction, *in* "25th Annual ACM Symposium on Theory of Computing," 1993, pp. 612–622.
9. T. Feder, Classification of homomorphisms to oriented cycles (draft), manuscript.
10. T. Feder, P. Hell, and J. Huang, List homomorphisms and circular arc graphs, submitted.
11. T. Feder, P. Hell, and J. Huang, List homomorphisms to general graphs, manuscript.
12. H. Fleischner and M. Stiebitz, A solution of a colouring problem of P. Erdős, *Discrete Math.* **101** (1992), 39–48.
13. M. R. Garey and D. S. Johnson, "Computers and Intractability," Freeman, San Francisco, 1979.
14. M. C. Golumbic, "Algorithmic Graph Theory and Perfect Graphs," Academic Press, New York, 1980.
15. W. Gutjahr, E. Welzl, and G. Woeginger, Polynomial graph colourings, *Discrete Appl. Math.* **35** (1992), 29–46.
16. P. Hell, Retractions de graphes, Ph.D. thesis, Université de Montreal, 1972.
17. P. Hell, Retracts in graphs, *in* "Graphs and Combinatorics," Lecture Notes in Math., Vol. 406, pp. 291–301, Springer-Verlag, New York/Berlin, 1974.
18. P. Hell and J. Nešetřil, On the complexity of H-colouring, *J. Combin. Theory Ser. B* **48** (1990), 92–110.
19. P. Hell and I. Rival, Retracts in graphs, *Canad. J. Math.* **39** (1987), 544–567.
20. P. Hell, J. Nešetřil, and X. Zhu, Duality and polynomial testing of tree homomorphisms, *Trans. Am. Math. Soc.* **348** (1996), 1281–1297.
21. M. Kubale, Some results concerning the complexity of restricted colourings of graphs, *Discrete Applied Math.* **36** (1992), 35–46.
22. R. Nowakowski and I. Rival, A fixed edge theorem for graphs with loops, *J. Graph Theory* **3** (1979), 339–350.
23. E. Pesch, "Retracts of Graphs," Athenaeum, Frankfurt am Main, 1988.
24. C. Thomassen, Every planar graph is five-choosable, *J. Comb. Theory Ser. B* **62** (1994), 180–181.
25. N. Vikas, Computational complexity of graph compaction, Ph.D. thesis, Simon Fraser University, 1997.
26. M. Voigt, List Colourings of planar graphs, *Discrete Math.* **120** (1993), 215–219.