ELSEVIER

Discrete Applied Mathematics 49 (1994) 95–105

DISCRETE APPLIED MATHEMATICS

The complexity of colouring symmetric relational systems

Richard Brewster*

Department of Mathematics and Statistics, Simon Fraser University, Burnaby, BC, Canada V5A 1S6

Received 29 March 1991; revised 28 January 1992

Abstract

A *relational system*, S, is a set T with relations $R_1, R_2, \ldots, R_k$ on T, denoted $S = (T, R_1, R_2, \ldots, R_k)$. We consider relational systems where all the relations are binary, symmetric and antireflexive. The *underlying graph*, G, of a relational system $S = (T, R_1, \ldots, R_k)$ is the graph with vertex set $V(G) = T$ and $uv \in (G)$ if $(u, v) \in R_i$ for some i. We use the terms *path* (respectively *tree*) to mean a relational system whose underlying graph is a path (respectively tree). Let $E_i(G) = \{uv | (u, v) \in R_i\}$ be the set of *edges coloured i*. The relational system defined by $(T, R_1, R_2, \ldots, R_k)$ can be represented in the form of a *coloured graph* $V(G)$ together with $E_1(G), E_2(G), \ldots, E_k(G)$. A *homomorphism* $G \to H$ is a mapping to $V(G)$ to $V(H)$ that preserves edge colours. Such a mapping is called an *H-colouring of G*. For a fixed H, the H-colouring problem is:

***H*-COL** (H-colouring).
Instance: A relational system G.
Question: Does there exist an H-colouring of G?

We show that H-COL is polynomial when H is a path and we show that trees exist such that H-COL is NP-complete. These results parallel the results of directed graphs concerning oriented paths and trees. However, the trees presented here are somewhat smaller and simpler than the known examples for directed graphs.

1. Introduction

A *relational system*, S, is a set T together with relations $R_1, R_2, \ldots, R_k$ on T, denoted $S = (T, R_1, R_2, \ldots, R_k)$. While relational systems are very general, we are interested in a generalization of graphs and therefore we only consider the case where all the relations are binary, symmetric and antireflexive. Of course, such a relational system

*The author wishes to thank the Natural Science and Engineering Research Council for its financial support.

SSDI 0166-218X(93)E0163-S

with only one relation is an undirected graph without loops. This motivates the following definition. The *underlying graph*, G, of a relational system $S = (T, R_1, \ldots, R_k)$ is the graph with vertex set $V(G) = T$ and $uv \in E(G)$ if $(u, v) \in R_i$ for some i. This allows us to talk about relational systems that are connected, bipartite, etc. by considering the underlying graph. In particular, we will use the term *path* (respectively *tree*) to mean a relational system whose underlying graph is a path (respectively tree). In addition to considering relational systems via their underlying graph, we may also wish to distinguish between edges from different relations. To this end, let $E_i(G) = \{uv: (u, v) \in R_i\}$ be the set of *edges coloured i*. Observe that $E(G) = E_1 \cup E_2 \cup \cdots \cup E_k$. The relational system defined by $(T, R_1, R_2, \ldots, R_k)$ will be presented in the form of a *coloured graph* $V(G)$ together with $E_1(G), E_2(G), \ldots, E_k(G)$.

Let G and H be two relational systems. A *homomorphism* from G to H, denoted $G \to H$, is a mapping $f: V(G) \to V(H)$ such that if $g_1 g_2 \in E_i(G)$, then $f(g_1) f(g_2) \in E_i(H)$ for $i = 1, 2, \ldots, k$. The relational system $f(G)$ has vertex set $V(f(G)) = \{h \in V(H)$: there exists $g \in V(G)$ such that $f(g) = h\}$ and edge sets $E_i(f(G)) = \{h_1 h_2 \in E_i(H)$: there exists $g_1 g_2 \in E_i(G)$ such that $f(g_1) f(g_2) = h_1 h_2\}$. A homomorphism, $G \to H$, is called an *H-colouring* of G. This is motivated by the fact that an n-colouring of a graph G is a homomorphism $G \to K_n$. We also write $G \to H$ to mean there exists a homomorphism from G to H. We are interested in the complexity of the following problem.

H-COL (H-colouring).
Instance: A relational system G.
Question: Does there exist an H-colouring of G?

For graphs, the complexity of H-COL has been completely determined in [5]. The problem is NP-complete if H is nonbipartite and polynomial otherwise. For directed graphs, such a nice characterization does not seem to exist. See [1, 2, 4]. In this paper we wish to study the complexity of H-COL for relational systems. In particular, we show the problem is polynomial when H is a path and there exist trees such that H-COL is NP-complete. These results are similar to the work of [4] where it is shown that H-COL is polynomial when H is an oriented path and there exists a directed tree for which H-COL is NP-complete. However, the trees presented here are somewhat smaller and somewhat simpler in that they are generalized stars. We have no particular application motivating this investigation other than this early work suggests the behaviour of H-colouring by relational systems can be very rich even in small examples.

A final definition we require is the *product* of G and H, denoted $G \times H$. It is the relational system on $V(G) \times V(H)$ where $(g_1, h_1)(g_2, h_2) \in E_i(G \times H)$ if and only if $g_1 g_2 \in E_i(G)$ and $h_1 h_2 \in E_i(H)$. Note that $G \times H \to G$ and $G \times H \to H$ via the projections ϕ_G and ϕ_H, where $\phi_G(g, h) = g$ and $\phi_H(g, h) = h$.

2. Path colourings

In this section we study the complexity of H-COL when H is a fixed path. There exists a polynomial algorithm to solve this problem. Our strategy is to show $G \to H$ if

and only if $G \to G \times H$. We solve the second problem using an algorithm similar to the one found in [4].

In the following, assume G is a relational system whose underlying graph is bipartite and H is a path. Note that if G is not bipartite, we can answer NO to H-COL, since a homomorphism $G \to H$ is also a homomorphism between the underlying graphs. We begin with a series of lemmas.

Lemma 2.1. *The relational system $G \times H$ has at least two components.*

Proof. Recall that both G and H are bipartite and we can assume the vertices of both G and H have been properly coloured with $\{1, 2\}$. If $(g_1, h_1)(g_2, h_2) \in E(G \times H)$, then g_1 and g_2 must be different colours in G and h_1 and h_2 must be different colours in H. Therefore, the only edges in $G \times H$ are between vertices coloured (1, 1) and vertices coloured (2, 2) or between vertices coloured (2, 1) and vertices coloured (1, 2). Observe that the subgraph induced by vertices coloured with $\{(1, 1), (2, 2)\}$ is not connected to the subgraph induced by vertices coloured with $\{(1, 2), (2, 1)\}$. □

Note it is possible to have more than two components in $G \times H$ even when G and H are connected relational systems as shown in Fig. 1. This differs from regular graphs, see [6, Proposition 1].

The algorithm in [4] solves H-COL when H is an oriented path. The algorithm in [4] requires that the target graph H have the so-called $\underline{X}$ property. The coloured graphs we use do not have the $\underline{X}$ property. Instead we use the following lemmas to show "crossing" edges in $G \times H$ are in different components.

Lemma 2.2. *The distinct edges $(g_1, h_1)(g_2, h_2)$ and $(g_1, h_2)(g_2, h_1)$ of $G \times H$ are in different components of $G \times H$.*

Proof. The vertices h_1 and h_2 are in different colour classes of any two-colouring of H. Similarly, g_1 and g_2 obtain different colours in any colouring of G. By the proof of Lemma 2.1, these edges are in different components. □

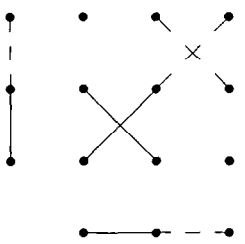

Fig. 1. A product of two connected relational systems with five components.

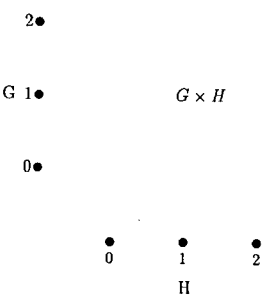

Fig. 2. The numbering of vertices in G and H.

For the following we will assume the vertices of G and H have been numbered as in Fig. 2.

Lemma 2.3. *Let $(g_1, h_1)(g_2, h_2)$ and $(g_1, h_3)(g_2, h_4)$ be two distinct edges in the same component in $G \times H$, then $h_1 \leqslant h_3$ and $h_2 \leqslant h_4$ or $h_1 \geqslant h_3$ and $h_2 \geqslant h_4$.*

Proof. We can assume without loss of generality that $h_1 \leqslant h_2$. Since H is a path and h_1h_2 is an edge of H, $h_2 = h_1 + 1$. Suppose $h_3 \leqslant h_4$. Again, since H is a path, $h_4 = h_3 + 1$. If $h_1 \leqslant h_3$, then $h_2 \leqslant h_4$. If $h_3 \leqslant h_1$, then $h_4 \leqslant h_2$. In either case the result holds. Now suppose $h_3 \geqslant h_4$. If $h_3 \leqslant h_1$ or $h_4 \geqslant h_2$, then the result is true. Hence, the only way for the lemma to fail is if $h_3 > h_1$ and $h_4 < h_2$. This implies $h_4 = h_1$ and $h_3 = h_2$. By Lemma 2.2 these edges are in different components contrary to our assumption. The result follows. □

Thus if we examine the depiction of $G \times H$, the edges between any two rows in some component of $G \times H$ have the property that no two edges "cross". Observe that if H is a tree this may not be the case, as demonstrated in Fig. 3. Note only one component of $G \times H$ is drawn. Here H is the tree while G is a single edge.

Lemma 2.4. *If G is connected and $f: G \to W$ is a homomorphism mapping G to some relational system W, then $f(G)$ is connected.*

Proof. Let u and v be two vertices in G. Since G is connected, there exists a path $u = p_0p_1 \ldots p_n = v$ such that $p_ip_{i+1} \in E(G)$ for $i = 0, 1, \ldots, n-1$. This implies $f(p_i)f(p_{i+1}) \in E(W)$ for $0 \leqslant i \leqslant n-1$, since f is a homomorphism. Hence $f(u) = f(p_0)f(p_1) \ldots f(p_n) = f(v)$ is a walk in W containing a path from $f(u)$ to $f(v)$. □

Lemma 2.5. *The relational system $G \to G \times H$ if and only if $G \to H$.*

Proof. If $G \to G \times H$, then since $G \times H \to H$ we have $G \to H$. On the other hand, suppose there exists a homomorphism $f: G \to H$. Let $\phi: G \to G \times H$ be the mapping defined by $\phi(g) = (g, f(g))$ for all $g \in V(G)$. Now if $g_1g_2 \in E_i(G)$, then $(g_1, f(g_1))(g_2, f(g_2)) \in E_i(G \times H)$. □

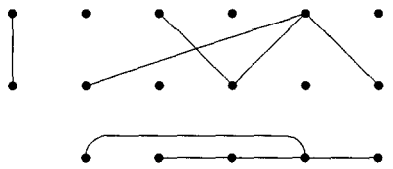

Fig. 3. A product with crossing edges.

By examining the map ϕ, we get the following corollary.

Corollary 2.6. *If $G \to H$, there exists a one to one homomorphism $G \to G \times H$.*

Proof. Using ϕ from above, if $\phi(g_1) = \phi(g_2)$, then $(g_1, f(g_1)) = (g_2, f(g_2))$. This implies $g_1 = g_2$. □

Using Lemmas 2.4 and 2.5 we get the following.

Corollary 2.7. *If G is connected, then $G \to H$ if and only if there is a one to one homomorphism from G to some connected component of $G \times H$.*

3. The path colouring algorithm

In this section we describe an algorithm and prove it solves H-COL in polynomial time when H is a path. We assume for the remainder of this section that G is connected. If G is not connected we can apply the algorithm on each component of G.

Let f_1 and f_2 be two homomorphisms from G to $G \times H$ of the form $f_i(g) = (g, h_j)$ for all $g \in V(G)$. That is, the same form as ϕ in Lemma 2.5. We say $f_1 \leqslant f_2$ if $f_1(g) \leqslant f_2(g)$ for all $g \in V(G)$, where the vertices of H are $\{0, 1, 2, \ldots, k\}$ as in Fig. 2. Here $f_1(g) = (g, h_1) \leqslant f_2(g) = (g, h_2)$ is equivalent to $h_1 \leqslant h_2$ since the first components are the same. A homomorphism $f: G \to G \times H$ is *minimum* if $f \leqslant f'$ for all homomorphisms $f': G \to G \times H$. We know from Corollary 2.7 that each homomorphism from G to H actually induces a one to one homomorphism from G into some component W of $G \times H$. Let W be a component of $G \times H$. We denote the set of homomorphisms $G \to W$ by $W(G)$.

Lemma 3.1. *Let H be a fixed path and G a relational system. For each component W of $G \times H$, either $W(G)$ is empty or $W(G)$ contains a minimum element.*

Proof. If $W(G)$ is empty for all components W of $G \times H$, then we are done. Suppose some $W(G)$ is not empty. Let f_1 and f_2 be two homomorphisms in $W(G)$ such that $f_1 \not\leqslant f_2$ and $f_2 \not\leqslant f_1$. Let $f_3(g) = \min\{f_1(g), f_2(g)\}$ for all $g \in V(G)$.

Claim. *The mapping $f_3: G \to W$ is a homomorphism.*

Suppose $g_1g_2 \in E_i(G)$. The pair $f_3(g_1)f_3(g_2)$ is (by definition of f_3) the pair $\min\{f_1(g_1), f_2(g_1)\}\min\{f_1(g_2), f_2(g_2)\}$.

We know that $f_1(g_1)f_1(g_2)$ and $f_2(g_1)f_2(g_2)$ are two edges in some component of $G \times H$. By Lemma 2.3, it must be the case that $f_1(g_1) \leqslant f_2(g_1)$ and $f_1(g_2) \leqslant f_2(g_2)$ or $f_1(g_1) \geqslant f_2(g_1)$ and $f_1(g_2) \geqslant f_2(g_2)$. In the first case $f_3(g_1) = f_1(g_1)$ and $f_3(g_2) = f_1(g_2)$. In the second case $f_3(g_1) = f_2(g_1)$ and $f_3(g_2) = f_2(g_2)$. Hence, $f_3(g_1)f_3(g_2)$ is an edge in W. This establishes the claim.

We conclude $W(G)$ must have a minimum element. □

Our aim is to describe an algorithm that finds a minimum homomorphism from G into a connected component of $G \times H$ and thereby solve H-COL in view of Corollary 2.7. We have two basic structures. Firstly, $\tilde{f}$ is a mapping from $V(G)$ to $V(H)$, which is not necessarily a homomorphism. Secondly, $\mathscr{C}$ is a subset of $E_1(G) \cup E_2(G) \cup \cdots \cup E_k(G)$. After choosing a component W of $G \times H$, we have the following two statements which are true throughout the algorithm.

(i) If $W(G)$ is not empty, then $\tilde{f} \leqslant f$ for all $f \in W(G)$.

(ii) If g_1g_2 is an edge in $E_\alpha(G) - \mathscr{C}$, then $\tilde{f}(g_1)\,\tilde{f}(g_2)$ is an edge of $E_\alpha(W)$ for all $\alpha \in \{1, 2, \ldots, k\}$.

We are now ready to describe the path colouring algorithm.

Step 1. Choose a component W in $G \times H$.

Step 2. Initially, set $\tilde{f} \equiv 0$ and $\mathscr{C} = E_1(G) \cup E_2(G) \cup \cdots \cup E_k(G)$.

Step 3. Choose an edge $g_1g_2 \in \mathscr{C}$. Let g_1g_2 be colour α.

Step 4. Choose the minimum (i, j) such that $(g_1, i)(g_2, j) \in E_\alpha(W)$ and $\tilde{f}(g_1) \leqslant (g_1, i)$ and $\tilde{f}(g_2) \leqslant (g_2, j)$. If no such (i, j) exists, then pick a new component and goto Step 2 or STOP and answer NO if all components have been tried.

Step 5. Update the colouring.

Step 5.1. If $\tilde{f}(g_1) = (g_1, i)$ and $\tilde{f}(g_2) = (g_2, j)$, then continue.

Step 5.2. If $\tilde{f}(g_1) \neq (g_1, i)$ and $\tilde{f}(g_2) = (g_2, j)$, then put all edges incident with g_1 into $\mathscr{C}$.

Step 5.3. If $\tilde{f}(g_1) = (g_1, i)$ and $\tilde{f}(g_2) \neq (g_2, j)$, then put all edges incident with g_2 into $\mathscr{C}$.

Step 5.4. If $\tilde{f}(g_1) \neq (g_1, i)$ and $\tilde{f}(g_2) \neq (g_2, j)$, then put all edges incident with g_1 and g_2 into $\mathscr{C}$.

Step 6. Set $\tilde{f}(g_1) = (g_1, i)$ and $\tilde{f}(g_2) = (g_2, j)$. Remove g_1g_2 from $\mathscr{C}$. If $\mathscr{C} = \emptyset$, then STOP and answer YES otherwise goto Step 3.

We need to show that the pair (i, j) in Step 4 is well defined. Suppose (i, j) and (m, n) are pairs of vertices in H such that $(g_1, i)(g_2, j) \in E(W)$ and $(g_1, m)(g_2, n) \in E(W)$, then by Lemma 2.3, either $(i, j) \leqslant (m, n)$ or $(m, n) \leqslant (i, j)$. Hence, a minimum does exist.

Theorem 3.2. *The path colouring algorithm solves* H-COL *in* $O(|V(G)| + |E(G)|)$ *time when* H *is a fixed path.*

Proof. We prove the algorithm works by induction on the number of edges checked. When zero edges have been checked, both invariants are trivially true. Suppose both

are true after n edges have been checked. Further suppose that the $(n+1)$st edge to be checked is g_1g_2. If $W(G)$ is empty, then invariant (i) is trivially true. If $W(G)$ is not empty, then let f be the minimum element of $W(G)$. We have by induction, $f(g_1) \geqslant \tilde{f}(g_1)$ and $f(g_2) \geqslant \tilde{f}(g_2)$. Also $f(g_1)f(g_2) \in E_\alpha(W)$ since f is a homomorphism. Therefore, at Step 4 the pair (i,j) exists with $f(g_1) \geqslant (g_1, i)$ and $f(g_2) \geqslant (g_2, j)$. The mapping $\tilde{f}$ is updated such that $\tilde{f}(g_1) = (g_1, i) \leqslant f(g_1)$ and $\tilde{f}(g_2) = (g_2, j) \leqslant f(g_2)$. By induction, $\tilde{f}(g) \leqslant f(g)$ for all $g \in V(G) - \{g_1, g_2\}$. Therefore, invariant (i) remains true.

Notice we have just proved if $W(G)$ is not empty, then the pair (i,j) exists at Step 4. Therefore, the algorithm only chooses a new component in $G \times H$ if the current $W(G)$ is empty. In order for the algorithm to reply NO, $W(G)$ must be empty for all components W.

If at Step 4 a new component is chosen, then returning to Step 2 makes both invariants trivially true again. If the pair (i,j) exists in Step 4, then $\tilde{f}$ is updated. By induction, invariant (ii) was true before $\tilde{f}$ was updated. The only edge removed from $\mathscr{C}$, and hence the only edge that could make invariant (ii) false, is g_1g_2. The choice of (i,j) at Step 4 guarantees that $\tilde{f}(g_1)\tilde{f}(g_2)$ is an edge coloured α. Therefore, invariant (ii) remains true. If $\mathscr{C}$ becomes empty upon removing g_1g_2, then $\tilde{f}$ is a homomorphism and the algorithm has correctly identified a YES instance.

Let $|V(H)| = p$. There are $|V(G)|p$ vertices in $G \times H$. For each edge in G there are at most $2p$ corresponding edges in $G \times H$. Therefore, $G \times H$ can be constructed in $O(|E(G)| + |V(G)|)$ time. Identifying the components of $G \times H$ requires $O(|E(G)|)$ time. Step 2 requires $O(|E(G)| + |V(G)|)$ time. An edge is added to $\mathscr{C}$ when one of its ends is increased. This means an edge can be added to $\mathscr{C}$ at most $2p-2$ times. Therefore, an edge can be checked at most $2p-1$ times. Choosing the minimum pair (i,j) in Step 4 requires constant time. Therefore, we require at most $(2p-1)|E(G)|$ iterations each of constant time. The total time required is $O(|V(G)| + |E(G)|)$. □

4. NP-complete trees

We now prove that there exists a tree H such that H-COL is NP-complete. Moreover, the two trees presented are generalized stars. Clearly, H-COL is in NP. Therefore, we need only provide a polynomial reduction from an NP-complete problem. The reduction is from ONE-IN-THREE 3SAT. See [3] for details on the complexity. Formally, ONE-IN-THREE 3SAT is defined below.

ONE-IN-THREE 3SAT.

Instance: Set U of variables, collection C of clauses over U such that each clause $c \in C$ has $|c| = 3$.

Question: Is there a truth assignment for U such that each clause in C has exactly one true literal?

This problem remains NP-complete if no $c \in C$ contains a negated literal.

Let H be the tree in Fig. 4. The edge colours are given by the letters beside each edge. For example, if the edge uv has abc beside it, then u is related to v in relations a, b, and c. In other words, under any homomorphism the preimage of uv is edges coloured

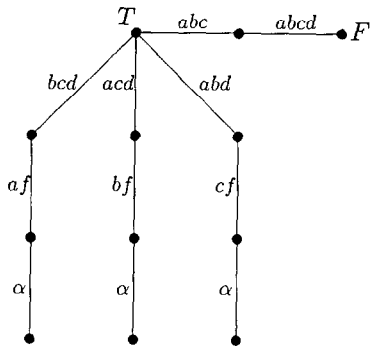

Fig. 4. An NP-complete generalized star.

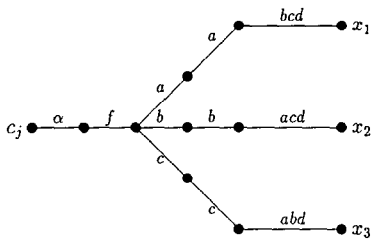

Fig. 5. Clauses for the tree.

with a, b or c. The idea of assigning different colours to the edges of H is similar to the use of super edges in [4] and hence the reduction is similar. However, the use of many colours allows us to construct smaller, simpler trees.

Theorem 4.1. *Let H be the tree in Fig. 4. Then H-COL is NP-complete.*

Proof. Let us be given an instance of ONE-IN-THREE 3SAT without negated variables. We construct a relational structure G. Let G have vertices $l_1, l_2, \ldots, l_m$, corresponding to the m literals in our instance of ONE-IN-THREE 3SAT. For each clause $c_j \in C$ with $c_j = l_{j_1} \vee l_{j_2} \vee l_{j_3}$, construct a copy of the structure in Fig. 5 with the vertices x_1, x_2 and x_3 identified with l_{j_1}, l_{j_2}, and l_{j_3}.

The structure G maps to H if and only if a truth assignment exists that assigns true to exactly one variable in each clause c_j. Suppose $G \to H$. It is easy to see that each vertex c_j gets mapped to one of the three leaves across the bottom of H, since c_j must get mapped to a vertex incident with an edge of colour α. Further, once c_j is mapped, the rest of the vertices in the clause have their colours forced.

Suppose c_j is mapped to the left-most leaf in Fig. 4. The edge α is mapped to α. The edge f must be mapped to f. The degree-four vertex in Fig. 5 must be mapped to the

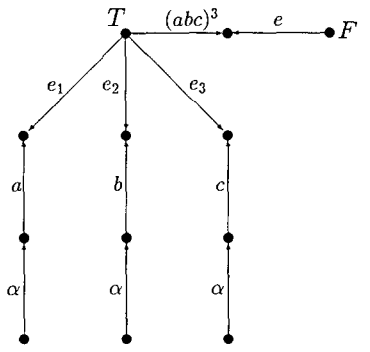

Fig. 6. An NP-complete 2-colour generalized star.

vertex in H incident with af below it and bcd above it. The branch in c_j consisting of a, a, bcd must map to a, a, bcd in H. This means x_1 is mapped to T. The branch b, b, acd must be mapped to $bcd, abc, abcd$. Thus, x_2 is mapped to F. Similarly, x_3 is mapped to F. If c_j is mapped to the middle leaf, x_2 is mapped to T while x_1, x_3 are mapped to F. Finally, if c_j is mapped to the right-most leaf, x_3 is true and x_1, x_2 are false. Hence, one of $\{x_1, x_2, x_3\}$ will be mapped to T and the other two will be mapped to F. This is the truth assignment that assigns true to one variable in each clause.

On the other hand, given a truth assignment, map all true literals to T and all false literals to F. This will again force the colouring of all vertices in G and produce a homomorphism. □

The above example is nice in that H contains only 12 vertices. The NP-complete directed tree found in [4] has 278 vertices. An example of a two-colour NP-complete tree exists on 98 vertices (see below). Perhaps allowing coloured edges let us observe richer behaviour in smaller examples.

Now we construct an NP-complete tree with two edge colours. Let H be the tree in Fig. 6. The labels on the edges here are not colours, but in fact refer to "super" edges. See Fig. 8. Each super edge consists of a sequence of blue edges, followed by a sequence of red edges, followed by a sequence a blue edges, followed by a single red edge. The number above each edge corresponds to the length of the sequence. For example, super edge a is a path of three blue edges, five red edges, five blue edges, and a single red edge. The super edge $(abc)^3$ has three edges in each sequence. The super edge $(abc)^5$ found in Fig. 7 has sequences of length five. Each super edge has an orientation from the white *super vertex* at one end to the black *super vertex* at the other. Each arc drawn in Figs. 6 and 7 is actually the corresponding oriented super edge. The super edge α (not shown) is an alternating path of length six. That is, α is obtained by adding a blue then a red edge to the black end of the super edge e. Note, e is an alternating path of length four.

We now tackle the somewhat cumbersome task of describing homomorphisms between the super edges. Consider the following proposition.

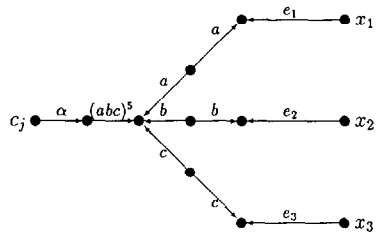

Fig. 7. Clauses for the 2-colour tree.

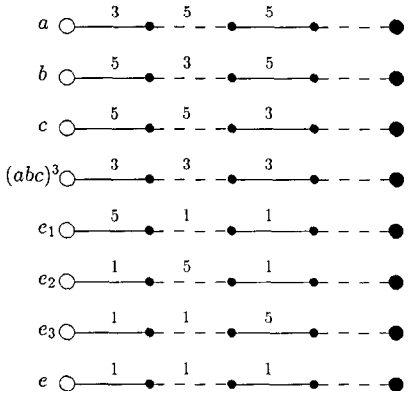

Fig. 8. Super edges.

Proposition 4.2. *Let* $P = p_0p_1 \ldots p_{2i+1}$ *and* $Q = q_0q_1 \ldots q_{2j+1}$ *be two paths with all edges blue. There exists a homomorphism,* f*, mapping* P *to* Q *such that* $f(p_0) = q_0$ *and* $f(p_{2i+1}) = q_{2j+1}$ *if and only if* $i \geqslant j$.

What does this mean in terms of super edges? Let P and Q be two super edges (neither of which is α). Suppose there exists a homomorphism from P to Q mapping the white (respectively black) super vertex of P to the white (respectively black) super vertex of Q. The initial sequence of blue edges in P must map onto the initial sequence of blue edges in Q with the ends in P mapping to the corresponding ends in Q. Also, both sequences have odd length. By Proposition 4.2, this can only occur if the sequence in P is at least as long as the sequence in Q. Now the second monochromatic sequences in P and in Q are red. Each sequence has odd length and corresponding ends must again map to each other. Therefore, the first red sequence in P must be at least as long as the first red sequence in Q. In other words, P maps to Q if and only if each monochromatic sequence in P is at least as long as the corresponding sequence in Q.

For example, the super edge a will map to the super edges $(abc)^3$, e_2, e_2, and e, but it will not map to the super edges b, c, or e_1. The super edge α will only map to an

alternating path of length six. If one checks the tree in Fig. 6, the only such paths are the three α super edges.

Theorem 4.3. *Let H be the tree in Fig. 6. Then H-COL is* NP*-complete.*

Proof. The proof works exactly the same as the previous tree. Suppose we are given an instance of ONE-IN-THREE 3SAT. We construct a graph G using the structure in Fig. 7 for each clause. Because the super edge α in each c_j only maps to one of the three super edges labeled α in H, the clauses map to H in the same way as described in Theorem 4.1. □

5. Conclusion

The complexity of H-COL for relational systems seems to have a lot in common with the complexity of H-COL for direct graphs in the cases when H is a path or a tree. This leads us to believe there will not be a nice characterization of H-COL as is the case with undirected graphs. However, coloured graphs seem to be an interesting tool for discovering behaviour about homomorphisms and may provide insight into existing problems involving graphs and directed graphs.

References

[1] J. Bang-Jensen and P. Hell, The effect of two cycles on the complexity of colourings by direct graphs, Discrete Appl. Math. 26 (1990) 1–23.
[2] J. Bang-Jensen, P. Hell and G. MacGillivray, The complexity of colouring by semi-complete digraphs, SIAM J. Discrete Math. 1 (1988) 281–298.
[3] M.R. Garey and D.S. Johnson, Computers and Intractability (Freeman, New York, 1979).
[4] W. Gutjhar, E. Welzl and G. Woeginger, Polynomial graph-colorings, Tech. Rept. B-88-06, University of Berlin.
[5] P. Hell and J. Nešetřil, On the complexity of H-colouring, J. Combin. Theory Ser. B 48 (1990) 92–110.
[6] D.J. Miller, The categorical products of graphs, Canad. J. Math. 20 (1968) 1511–1521.