List Partitions and Trigraph Homomorphisms

Tomás Feder
tomas@theory.stanford.edu,
Pavol Hell
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6
pavol@cs.sfu.ca,
and
Kim Tucker-Nally
Mathematics Department
Capilano College
2055 Purcell Way
North Vancouver, B.C. V7J 3H5
knally@capcollege.bc.ca

October 10, 2003

Abstract

Trigraph homomorphisms are useful to model certain graph partition problems. We classify the complexity of all list homomorphisms to directed trigraphs with up to three vertices.

1 Introduction

A *trigraph* G consists of a set $V = V(G)$ of vertices, and of two disjoint binary relations on V – the *weak* relation $W = W(G)$, and the *strong* relation $S = S(G)$. Thus each relation is a directed graph on V, and we use the usual digraph terminology and notation with respect to arcs, loops, etc. (The fact that the relations are disjoint means that no arc uv is in both relations.) A trigraph is by definition directed, but if both relations W and S are symmetric, we can view each symmetric pair of arcs uv, vu as an undirected edge uv, and obtain an *undirected trigraph*. When we depict trigraphs in figures, we distinguish weak arcs from strong arcs by the width of the lines. (The strong arcs are thicker.)

The *adjacency matrix* of a trigraph G (with respect to a fixed enumeration of vertices), is the $n \times n$ matrix M over $0, *, 1$, in which $M(u, v) = 0$ if uv is not an arc, $M(u, v) = *$ if uv is a weak arc ($uv \in W(G)$), and $M(u, v) = 1$ if uv is a strong arc ($uv \in S(G)$). Note that a trigraph is undirected if and only if its adjacency matrix is symmetric.

Let G and H be trigraphs. A *homomorphism* of G to H is a mapping $f : V(G) \to V(H)$ such that the following three conditions are satisfied:

- $uv \in W(G)$ implies $f(u)f(v) \in W(H)$
- $uv \in S(G)$ implies $f(u)f(v) \in W(H) \cup S(H)$
- $uv \notin W(G) \cup S(G)$ implies $f(u)f(v) \notin S(H)$.

We note that for each trigraph G the identity mapping of $V(G)$ to $V(G)$ is a homomorphism. Moreover, if G, H, K are trigraphs, f a homomorphism of G to H, and g a homomorphism of H to K, then the composition $g \circ f$ is a homomorphism of G to K.

Most frequently we will be discussing homomorphisms of G to H when G is a (directed) graph. A trigraph G with $W(G) = \emptyset$ is called a *strong graph* and a trigraph H with $S(H) = \emptyset$ is called a *weak graph*. Suppose $f : G \to H$ is a homomorphism: If G is a graph we always assume it is a strong graph; if H is a graph we always assume it is a weak graph. Note that a homomorphism of a strong graph to a weak graph as defined above conforms to the usual definition of graph homomorphism.

Additional intuition about the role of the weak and the strong edges can be derived from the following alternate definition of trigraphs: A trigraph G is a set $V(G)$ of vertices and two binary relations $E(G), N(G)$ on $V(G)$, such that $E(G) \cup N(G) = V(G) \times V(G)$. A homomorphism of a trigraph G to a trigraph H is a mapping $f : V(G) \to V(H)$ such that $gg' \in E(G)$ implies $f(g)f(g') \in E(H)$ and $gg' \in N(G)$ implies $f(g)f(g') \in N(H)$. A strong graph is a trigraph G in which $E(G) \cap N(G) = \emptyset$. A weak graph is a trigraph G in which $N(G) = V(G) \times V(G)$. The alternate definitions are equivalent, as is easily seen by letting $S(G) = E(G) - N(G)$ and $W(G) = E(G) \cap N(G)$, and, in the other direction, $E(G) = S(G) \cup W(G)$ and $N(G) = (V(G) \times V(G)) - S(G)$.

Following standard practice, we also call a homomorphism of G to H an *H-colouring of G*. Suppose H is a fixed trigraph, and suppose G is a trigraph in which each vertex $v \in V(G)$ has a *list* $L(v) \subseteq V(H)$. A *list*

H-*colouring* of G, with respect to the lists L, is a homomorphism f of G to H such that $f(v) \in L(v)$ for all $v \in V(G)$.

Let H be a fixed trigraph. The H-*colouring problem* asks whether or not an input (strong) graph G admits an H-colouring. The **List H-colouring Problem** takes as input a (strong) graph G with lists L, and asks whether or not G admits a list H-colouring with respect to L. Note that the H-colouring problem is a restriction of the list H-colouring problem to instances G, L where each $L(g)$ equals $V(H)$.

We emphasize that an instance of the list H-colouring problem is a *graph* G. For each fixed trigraph H, the list H-colouring problem is about vertex partitions of graphs G. For instance, if H is the undirected trigraph with $V(H) = \{0, 1\}$, with a strong loop on 1 and a weak edge joining the two vertices $0, 1$, then an H-colouring of G is precisely a partition of $V(G)$ into a clique and an independent set. Thus G is H-colourable if and only if G is a split graph. Many examples of important graph partition problems, especially those occuring in the theory of graph perfection, can be formulated as trigraph homomorphism (or list homomorphism) problems [4].

The *complement* of a trigraph H is the trigraph whose adjacency matrix is obtained from the adjacency matrix of H by replacing each 0 by a 1 and each 1 by a 0. It is clear that if H' is the complement of H, then the list H'-colouring problem is polynomially equivalent to the list H-colouring problem.

In [5] we proved that for each trigraph H, the list H-colouring problem is NP-complete or *quasipolynomial* (of complexity $n^{O(log n)}$). Bulatov [2] has recently shown that for each weak graph H, the list H-colouring problem is NP-complete or polynomial. For all undirected trigraphs H on three or fewer vertices, the list H-colouring problem is polynomial time solvable, except for the problem of 3-colourability and the problem of stable cutset [4]. In this paper we extend this classification to all (directed) trigraphs H with up to three vertices. There are a few more NP-complete cases (cf. Fig.1), and all other list H-colouring problems are shown polynomial time solvable. We note that for trigraphs H with four vertices there are problems whose complexity is an open problem, even when H is undirected [3]. In fact [3], there is a unique undirected trigraph H for which the complexity of the list H-colouring problems has not been determined; this problem has been dubbed the "stubborn problem" by the authors of [3].

Most of these results have been first reported in the last author's M.Sc. thesis [12].

2 Basic Polynomial Time Algorithms

In [4] we have developed several techniques for solving list H-colouring problems for undirected trigraphs H. Many of those techniques apply to directed trigraphs H as well.

For instance, when H is a trigraph with at most two vertices, all list H-colouring problems can be solved by reducing the problem to 2-satisfiability, as described in [4]. The same applies to any list H-colouring problem in which every list has at most two elements.

Another technique that adapts well to directed trigraphs concerns domination: We say that vertex u *dominates* vertex v in a trigraph H with adjacency matrix M, if for each vertex x of H (including u and v), the entries $M(u,x), M(v,x)$ are the same or $M(u,x) = *$, and the entries $M(x,u), M(x,v)$ are the same or $M(x,u) = *$. If u dominates v in H, then we may delete v from any list containing u. If H has three elements, this reduces all lists to size at most two and hence the list H-colouring problem can be solved in polynomial time by 2-satisfiablity.

Suppose the adjacency matrix M of a trigraph H has a row or a column containing both a 0 and a 1. Without loss of generality, suppose that $xy \notin W(H) \cup S(H)$, and $xz \in S(H)$ (where either y or z could equal x). Then the list H-colouring problem for a graph G with n vertices reduces to the following $n+1$ problems: One problem obtained by deleting x from all lists (corresponding to an H-colouring of G in which no vertex is coloured by x), and n problems obtained by choosing a particular vertex g of G, setting $L(g) = x$, deleting y from all lists $L(g')$ where $gg' \in S(G)$ and deleting z from all lists $L(g')$ where $gg' \notin S(G)$. (These problems correspond to having the particular vertex g coloured by x.) As noted in [4] for undirected trigraphs, if a trigraph H has only three vertices, then all the reduced problems have lists of size at most two, and hence this gives a polynomial time algorithm for the list H-colouring problem, when the adjacency matrix of H has a row or a column with both a 0 and a 1.

If H is a weak graph ($S(H) = \emptyset$), we employ two techniques from [7, 10]. (More are available [2], but we had no need for them in this context.)

A *conservative majority function* μ on H is a mapping $V(H) \times V(H) \times V(H) \to V(H)$ such that

1. $\mu(u, v, w) \in \{u, v, w\}$,
2. if two of u, v, w are equal z then $\mu(u, v, w) = z$, and
3. if $uu', vv', ww' \in W(H)$ then $\mu(u, v, w)\mu(u', v', w') \in W(H)$.

It is proved in [7, 6] that if a weak graph H admits a conservative majority function, then the list H-colouring problem is solvable in polynomial time.

An *X-underbar enumeration* of a weak graph H is a total order on its vertices such that if uv and $u'v'$ are in $W(H)$ then $min(u, u')min(v, v')$ is also in $W(H)$. It is shown in [10] (cf. also [6]) that if a weak graph H admits an X-underbar enumeration, then the list H-colouring problem is solvable in polynomial time. We remark that, in the terminology of [7], a list H-colouring problem in which H has an X-underbar enumeration is *of width one*, while a list H-colouring problem in which H has a conservative majority function is *of strict width two.*

3 Weak Graphs H

We first focus on the case where H is a weak graph. Recall that a general classification of the complexity of list H-colouring problems as NP-complete and polynomial time solvable has been given by Bulatov [2]. (It does not, however, yield a graph theoretic description of the easy or hard cases.) We prove the following classification:

Theorem 3.1 *Let H be a weak graph with three vertices.*

- *If H, or its complement, is isomorphic to one of the graphs in Fig.1, then the list H-colouring problem is NP-complete.*
- *If H, or its complement, has at most two loops and is not isomorphic to one of the graphs in Fig.1 or Fig.3, then H has a conservative majority function.*
- *If H is one of the graphs in Fig.3, or if H has three loops, then H has an X-underbar enumeration.*

Lemma 3.2 *For all graphs H in Fig.1, the list H-colouring problem is NP-complete.*

Proof. Fig.1(a) and (b) yield the well-known NP-complete problems of (list) three-colouring and (list) stable cutset, cf. [4]. Fig.1(c) and (d) are also known to yield NP-complete H-colouring problems, even without lists. Indeed, it is proved in [1], that if the underlying undirected graph of H is complete, then the presence of two directed cycles in H is both necessary and

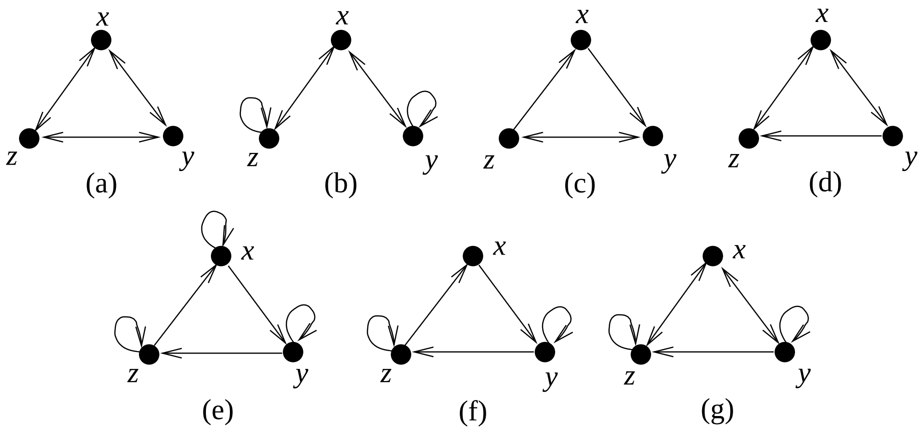

Figure 1: Weak graphs H with NP-complete list H-colouring problems.

sufficient for NP-completeness of the H-colouring problem (even without lists).

The graph H in Fig.1(g) has only one arc that is not symmetric - the arc yz. When yz is deleted, we have the graph in Fig.1(b), corresponding to the problem of list stable cutset, well-known to be NP-complete even for symmetric graphs [4]. Thus given a symmetric graph instance G, L of the list stable cutset problem, we see that G admits a list H-colouring with respect to L if and only if it admits a list stable cutset partition with respect to L. Thus list H-colouring is also NP-complete.

Let now H be the graph in Fig.1(f). We shall apply the indicator construction from Lemma 1 of [8]. Let I be the graph with vertices i, k, j and arcs ik, kk, kj. Let H^* be the graph on the vertices of H defined as follows: uv is an arc of H^* if and only if there exists a homomorphism of I to H which takes i to u and j to v. Given an instance G, L of list H^*-colouring, we can replace each arc gg' of G by a copy of I, identifying i with g and j with g'. Call the new graph ${}^*G$. The lists in ${}^*G$ of the vertices of G remain the same as in G, and the new vertices have lists $V(H)$. It is then easy to see that, as in [8] (where this construction is introduced without lists), that ${}^*G$ has a list homomorphism to H if and only if G has a list homomorphism to H^* (with respect to the lists L, in the case of ${}^*G$ amended as described above). Thus the list H^*-colouring problem is reduced (in polynomial time) to the list H-colouring problem. In our case, it is a simple exercise to check that H^* is the graph in Fig.1(g), proved NP-complete in the previous paragraph.

Finally, we prove that list H-colouring in NP-complete for the graph H in Fig.1(e). Consider the gadget in Fig.2 below. The lists of the vertices of the central three-cycle are marked; all other lists are $\{x, y, z\}$.

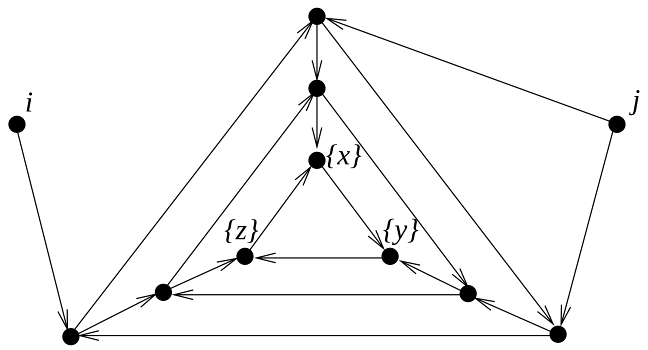

Figure 2: The gadget

In a list H-colouring of the gadget, the colours of the central three vertices are prescribed. They imply that the second three-cycle can only be coloured in one of two ways, x, y, z or z, x, y, going clockwise from the top. This in turn implies that the outermost three-cycle can only be coloured x, y, z or z, x, y or y, z, x going clockwise from the top. In the first case, the possible combinations of colours for the vertices i and j are y, x or z, x respectively; in the second case, they are y, z or x, z respectively; and in the last case z, y or x, y respectively. Note that these are all possible ordered pairs of distinct colours from x, y, z. Hence we can reduce the problem of three-colourability of an undirected graph G to the problem of list H-colouring the graph G' obtained from G by replacing each undirected edge uv of G with a copy of the gadget, identifying u with i and v with j. The above remarks imply that G is three colourable if and only if G' has a list H-colouring. $\square$

We now proceed to prove that all other problems are polynomial time solvable. We first construct conservative majority functions. Note that to define such a function μ on a graph H with three vertices x, y, z, we only need to define the values $\mu(x, y, z), \mu(x, z, y), \ldots, \mu(z, y, x)$, for the permutations of x, y, z. (The values that involve repeated vertices are prescribed by the definition of majority functions.) For the most part, we will define all these values to be equal to the same vertex $\mu(x, y, z) = \mu(x, z, y) = \cdots = \mu(z, y, x) = \mu$.

Lemma 3.3 *Suppose H has at most one loop. If both H and its complement are nonisomorphic to any of the graphs in Fig.1, then H admits a conservative majority function.*

Proof. Suppose first that H has no loops. If some vertex, say x, has no out-arcs then letting $\mu = x$ defines a conservative majority function on H. Indeed, $xa \in W(H)$ is not possible, so it only remains to check what happens when $ax \in W(H), by \in W(H), cz \in W(H)$, for some vertices a, b, c of H. Note that a, b, c are not distinct, since none of them can be x. Thus $\mu(a, b, c) = \mu(b, c, a) = \ldots \mu(c, b, a) = v$ where v is either y or z (whichever occurs at least twice amongst a, b, c). If $v = a$ then $\mu(a, b, c)\mu(x, y, z) \in W(H)$ (and similarly for other permutations of the vertices). Otherwise $b = c$ is either y or z, contrary to $by, cz \in W(H)$. A similar definition applies if some vertex has no in-arcs. Thus it remains to discuss graphs H in which each vertex has positive indegree and positive outdegree. If such an H contains a directed three-cycle, say xyz, and at least one other arc, then we have the situations in Fig.1(c,d). If H is just the directed three-cycle xyz, then it admits a conservative majority function μ which has $\mu(a, b, c) = a$ for any three distinct vertices a, b, c. (This is the only case where our conservative majority function is sensitive to the order of the arguments.) If a graph H in which all vertices have positive in- and outdegree does not contain a directed three-cycle, then it must be isomorphic to the symmetric graph H with arcs xy, yx, xz, zx. It is easy to check that setting $\mu = z$ defines a conservative majority function for this graph H.

If H has a unique loop, say at vertex x, then setting $\mu = x$ always defines a conservative majority function. Indeed, suppose that xa, yb, zc are (weak) arcs of H; we wish to show that $xv \in W(H)$ for $v = \mu(a, b, c)$. This is clear if $v = x$. Otherwise, as above, $b = c = v \neq x$, contradicting the fact that neither y nor z has a loop. $\square$

Lemma 3.4 *Suppose H has two loops. If both H and its complement are nonisomorphic to any of the graphs in Fig.1 and Fig.3, then H admits a conservative majority function.*

Proof. Suppose H has loops at y and z.

If both $yz \in W(H)$ and $zy \in W(H)$, then setting $\mu = y$ defines a conservative majority function, since $xa, yb, zc \in W(H)$ with $yv \notin W(H), v = \mu(a, b, c)$, would mean that $v = x$ and hence two of a, b, c are equal to x, contary to the fact that $xx \notin W(H)$ and $yx \notin W(H)$. (A similar argument applies to $ax, by, cz \in W(H)$.)

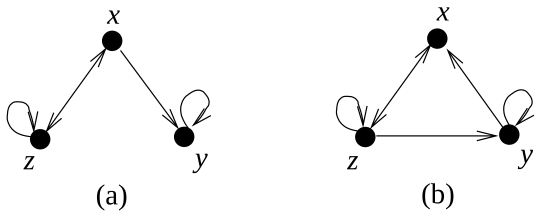

Figure 3: Two graphs without a conservative majority function.

Suppose next that only one of yz, zy is an arc of H, say $yz \in W(H)$. If $zx \notin W(H)$, we may define $\mu = y$ and obtain a conservative majority function. Indeed, $xa, yb, zc \in W(H)$ with $yv \notin W(H), v = \mu(a, b, c)$, is again impossible because it would mean that $v = x$ and hence two of a, b, c are x contrary to $xx, zx \notin W(H)$. Similarly, $ax, by, cz \in W(H)$ with $vy \notin W(H)$ would mean that $v = x$ or $v = z$: If $v = x$ then two of a, b, c must be x, yet neither a nor b can be x. If $v = z$ then two of a, b, c must be z, yet neither a nor b can be z. If $xy \notin W(H)$, we may similarly define $\mu = z$. In the remaining case, $xy \in W(H), zx \in W(H)$, we obtain the graphs H in Fig. 1(f, g) and Fig.3(b).

If neither yz nor zy is an arc of H, and H is not the graph in Fig.1(b) or Fig.3(a), then, up to symmetry, we may assume that either xz and xy are not arcs of H, or xz and yx are not arcs of H. We may assume that H is connected, otherwise a conservative majority function for H is easily defined by setting $\mu = y$ or $\mu = z$. Thus we may assume that $xy \in W(H)$, and either xz or zx is in $W(H)$. In either case, it is easy to see that setting $\mu = x$ defines a conservative majority function. □

To complete the proof of the theorem, it remains to consider the graphs H from Fig.3, and graphs with three loops:

Proof. The case of graphs in which every vertex has a loop was investigated in [6]. Two theorems from [6] are useful: The first one deals with graphs H whose underlying undirected graph is complete. The second one deals with the case when the underlying undirected graph is a tree. In both cases, [6] implies that if H has three vertices, then H admits an X-underbar enumeration. The only graphs H not covered by these results are disconnected graphs. It is obvious that any disconnected graph on three vertices admits an X-underbar enumeration (in which the components of H are enumerated one by one).

We complete the proof by noting that both graphs from Fig.3 admit an X-underbar enumeration, namely, z, x, y. □

Corollary 3.5 *If H is a graph such that both H and its complement are nonisomorphic to the graphs in Fig.1, then list H-colouring is solvable in polynomial time.*

4 Proper Trigraphs

The previous section deals with trigraphs whose adjacency matrix has no 1's. By complementation, we obtain a classification for those trigraphs whose adjacency matrix has no 0's. A trigraph is called *proper* if both it and its complement contain a strong arc (which could be a loop).

In this section we deal with proper trigraphs on three vertices. Thus we assume that the adjacency matrix M contains both a 0 and a 1. As observed in Section 2, if some 0 and 1 occur in the same row or column, we have a polynomial time algorithm.

We also have a generic polynomial time algorithm in the following case:

Theorem 4.1 *Suppose H is a trigraph with three vertices x, y, z, where xx is a strong loop and yz is not an arc. Then list H-colouring is polynomial time solvable.*

Proof. Consider a homomorphism f of a strong graph G to H. Let G' be the undirected graph on $V(G)$ consisting of edges gg' such that $gg' \in S(G)$ and $g'g \in S(G)$. Observe that f induces a vertex partition of G' into the clique $C = f^{-1}(x)$ and the sets $A = f^{-1}(y), B = f^{-1}(z)$, which are not joined by any edge of G'. According to Theorem 4.4 of [4] there exists in G' a family of $|V(G')|$ sets E_i such that whenever the vertices of G' are partitioned into a clique C and a pair of sets A, B with no edges of G' joining them, some set E_i contains C and is disjoint from A or from B. Moreover, this family can be found in polynomial time [4].

Our algorithm for solving the list H-colouring problem is the following: Given an input graph G with lists L, we form the undirected graph G' and generate the family of sets E_i. For each set E_i we generate two problems: In both problems x is removed from the lists of all vertices not in E_i. In the first problem, y is removed from the lists of all vertices in E_i, and in the second problem, z is removed from the lists of all vertices in E_i. It follows that G has a list H-colouring with respect to the given lists L if and only if in at least one of these $2|V(G)|$ auxiliary problems G has a list H-colouring with respect the modified lists. Since the modified lists all have size at most two, this can be checked in polynomial time. $\square$

By symmetry and complementation, we have polynomial time algorithms in all cases where one of $0, 1$ appears on the main diagonal and the other appears in an off-diagonal position.

For matrices where both a 0 and a 1 appear in diagonal positions, we can apply the following result:

Theorem 4.2 *Suppose H is a trigraph with three vertices x, y, z, where xx is a strong loop and yy is not a loop. Then list H-colouring is polynomial time solvable.*

Proof. By assumption, the adjacency matrix of H has a diagonal 0 and a diagonal 1. According to Theorem 4.1 and the remarks in Section 2, we only need to consider cases where there isn't a 0 and 1 in the same row or column, or an addidional 0 or 1 off the main diagonal. This implies that xy, xz, yx, yz, zx, and zy are all weak arcs; in other words all off-diagonal entries in the adjacency matrix of H are $*$. Thus the adjacency matrix is symmetric, and the problems have been solved by polynomial algorithms in [4]. (If zz is not a loop, or is a strong loop, then the list H-colouring problem is known as the problem of split $(1, 2)-$ or $(2, 1)-$ partitions; if zz is a weak loop, then z dominates both x and y.) $\square$

It remains to consider the case when the adjacency matrix M of H has a 0 and a 1 but only in off-diagonal positions. Thus we may assume that the diagonal has only $*$'s:

Theorem 4.3 *If H is a proper trigraph with three weak loops, nonisomorphic to any of the trigraphs in Fig.4, then the list H-colouring problem is polynomial time solvable.*

Proof. We assume that M contains no row or column with both a 0 and 1, otherwise the conclusion follows from the remarks in Section 2. Suppose first that there exists a strong arc ab such that ba is not an arc, for some a, b. Without loss of generality, assume that $M(x, y) = 1$ and $M(y, x) = 0$. Consider the values of $M(y, z)$ and $M(z, x)$. By our assumption, neither value can be 1. If $M(y, z) = M(z, x) = 0$, then $M(x, z) = M(z, y) = *$, yielding the first trigraph in Fig.4. In all the remaining cases ($M(y, z) =$

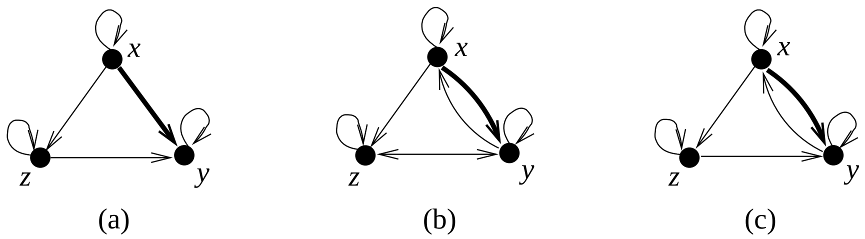

Figure 4: Three exceptional trigraphs.

$M(z,x) = *$; $M(y,z) = 0, M(z,x) = *$; and $M(y,z) = *, M(z,x) = 0$), it is easy to see that z must dominate x or y.

If the reverse of each strong arc is a (weak or strong) arc, then we may assume without loss of generality that $M(x,y) = 1$ and $M(z,x) = 0$. This implies that $M(z,y) = *$ and also that $M(y,x) = M(x,z) = *$ (since the reverse of each strong arc is an arc). The trigraph with $M(y,z) = 0$ is isomorphic to the complement of the trigraph with $M(y,z) = 1$, and is given in Fig.4(c). The last case, with $M(y,z) = *$ given in Fig.4(b). $\square$

We first describe polynomial time algorithms for the trigraphs in Fig.4(a,b). Recall the alternate view of trigraphs whereby the weak arcs correspond to relations E and N and the strong arcs only to relation E. Enumerate the vertices of the trigraph H in Fig.4(a) as x, z, y; it is easy to check that this is an X-underbar enumeration for both relations E and N, and hence the list H-homomorphism problem is polynomial time solvable, even if the inputs are any sets $V(G)$ with two binary relations E and N (i.e., not necessarily restricted to those instances G where E and N partition $V(G) \times V(G)$). For the graph in Fig.4(b) such an enumeration is z, y, x.

It remains to give a polynomial time algorithm to solve the list H-colouring problem for the third trigraph from Fig.4:

Theorem 4.4 *If H is the trigraph in Fig.4(c), then the list H-colouring problem is polynomial time solvable.*

Proof.

Let G, L be an instance of the list H-colouring problem. We let G' be the subgraph of G induced by the vertices with lists of size three; let n be the number of vertices of G' (the *size* of G'). We shall say that a vertex of G' has *low outdegree* if it has outdegree at most $n/2$; otherwise we shall say that it has *high outdegree*. Similarly, we shall say that a vertex of G' has *low indegree* if it has indegree at most $n/2$; otherwise we shall say that it has *high indegree.*

We shall describe a recursive algorithm, in which the graph G' and its size will vary. Consider the following four situations:

(0) A vertex v in G' has high indegree and high outdegree. Then if v is mapped to y or z, the size of G' is reduced by half, since no outneighbour of v can be mapped to z or x, respectively. If v is mapped to x, then the size of G' is also reduced by half since, similarly, no inneighbours of v can be mapped to z. Thus we reduce the problem to three subproblems with the size n of G' reduced by half.

(1) A vertex v in G' has high indegree and low outdegree. Then we would have two cases with n reduced by half, but no reduction if v is mapped to y.

(2) A vertex v in G' has low indegree and high outdegree. Then we would have two cases with n reduced by half, but no reduction if v maps to x.

(3) A vertex v in G' has low indegree and low outdegree. Then we would have two cases with n reduced by half, but no reduction if v maps to z.

As long as we can find a vertex v in situation (0), we can replace the current problem with three subproblems with half the size n. When such a vertex does not exist, we proceed as follows: We first attempt to map all vertices from (1) to y, all vertices from (2) to x, and all vertices from (3) to z. Since this is at this point an unjustified choice, we call these assignments *auxiliary.* Solve the resulting instance by 2-satisfiability. If a solution is found then we are done. Otherwise, the failure to find a solution to the instance of 2-satisfiability can be blamed on one or two of these auxiliary assignments. Indeed, it is well known that if an instance of 2-satisfiablity has no solution, then there is either a cycle of implications involving one variable and its negation, or a path of implications connecting two single-literal clauses ([11], Problem 6, Chapter 15.). If no auxiliary assignments are implicated by such a path or cycle, then there is no solution of the corresponding subproblem even if the auxiliary assignments are not made. Otherwise, one or two auxiliary assignments need to be changed. If it is only one assignment, say that of v to x in case (2), we obtain two subproblems (v maps to y or to z), with the size n halved. If two auxiliary assignments need changing, say v in case (1) and v' in case (2) (the other combinations are similar), then we cannot simultaneously map v to y and v' to x. This gives four cases that reduce n by half, namely, either v maps to z and n is halved because of the high indegree of v, or v maps to x and n is halved because of the high indegree of v, or v' maps to y and n is halved because of the high outdegree of v', or v' maps to z and n is halved because of the high outdegree of v'.

Thus in each of the combinations for v, v' (namely (1)-(1), (1)-(2), (1)-(3), (2)-(2), (2)-(3), (3)-(3)), we are left with four subproblems with n reduced by half. Iterating this transformation generates $4^{log_2 n} = n^2$ cases with G' empty, that is, all lists of size at most 2, which can be solved by 2-satisfiability. $\square$

Corollary 4.5 *Each proper trigraph H with at most three vertices has a polynomial time solvable H-colouring problem.*

5 Future Directions

A *k-graph* H consists of a set $V(H)$ of vertices, and k binary relations $E_1, E_2, \ldots, E_k$ such that $\cup_{i=1}^{k} E_i = V(H) \times V(H)$. A k-graph is *simple* if the relations E_i are disjoint, i.e., partition $V(H) \times V(H)$. A homomorphism of a k-graph G to a k-graph H is a mapping of $V(G)$ to $V(H)$ which preserves all the relations E_i. Thus 2-graphs are precisely trigraphs, and simple 2-graphs are ordinary graphs ($E_1 = E, E_2 = N$).

Let H be a fixed k-graph. The list H-colouring problem takes as input a simple k-graph G with lists L (each $L(g)$ is a subset of $V(H)$), and asks whether or not there exists a list homomorphism of G to H with respect to L. These problems have been studied in [5].

Here we focus on just one case, a generalization of split graphs. Let H_k be the symmetric k-graph with vertices $0, 1, \ldots, k-1$ in which each E_i equals $V(H) \times V(H) - ii$. Then an H_k-colouring of a simple k-graph G is a colouring of the vertices $V(G)$ by colours $0, 1, \ldots, k-1$, so that an edge in E_i does not have both endpoints coloured i.

Theorem 5.1 *([5] When $k = 2$, the list H_k-colouring problem is polynomial time solvable. When $k = 3$, the list H_k-colouring problem is solvable in quasipolynomial time. When $k \geq 4$, the list H_k-colouring problem is NP-complete.*

We do not know whether or not the H_3-colouring problem can be solved in polynomial time, even without lists.

References

[1] J. Bang-Jansen, P. Hell and G. MacGillivray, The complexity of colourings by semicomplete digraphs, SIAM J. Discrete Math. 1 (1988) 281–289.

[2] A.A. Bulatov, Tractable conservative constraint satisfaction problems, Tech Report Oxford University Computing Laboratory, PRG-RR-03-01.

[3] K. Cameron, E.M. Eschen, C. Hoang, and R. Sritharan, The complexity of the list partition problem for graphs, SODA 2004.

[4] T. Feder, P. Hell, S. Klein, and R. Motwani, List partitions, SIAM J. Comput., in press.

[5] T. Feder and P. Hell, List constraint satisfaction and list partition, manuscript.

[6] T. Feder, P. Hell, and J. Huang, List homomorphisms to reflexive digraphs, manuscript.

[7] T. Feder, M.Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction, SIAM J. Comput. 28 (1998) 57 - 104.

[8] P. Hell, J. Nešetřil, On the complexity of H-coloring, J. Combin. Theory Ser. B 48 (1990) 92–110.

[9] P. Hell, Algorithmic aspects of graph homomorphisms, BCC 2003.

[10] H.A. Maurer, J.H. Sudborough and E. Welzl, On the complexity of the general colouring problem, Inform. and Control 51 (1981) 123–145.

[11] C.H Papadimitriou, K. Steiglitz, **Combinatorial Optimization**, Prentice Hall, 1982.

[12] K. Tucker-Nally, List M-partitions of digraphs, M.Sc. Thesis, Simon Fraser University, School of Computing Science, 2003.