MATRIX PARTITIONS OF DIGRAPHS

by

David George Schell

Hon. B.Sc. Wilfrid Laurier University, Department of Physics & Computer Science, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2008

APPROVAL

Name: David George Schell

Degree: Master of Science

Title of thesis: Matrix Partitions of Digraphs

Examining Committee: Dr. Joseph Peters
Chair

Dr. Pavol Hell, Senior Supervisor

Dr. Andrei Bulatov, Supervisor

Dr. Ladislav Stacho, Examiner

Date Approved: _____

Abstract

The matrix partition problem has been of recent interest in graph theory. Matrix partitions generalize the study of graph colourings and homomorphisms. Many well-known graph partition problems can be stated in terms of matrices. For example skew partitions, split partitions, homogeneous sets, clique-cutsets, stable-cutsets and k-colourings can all be modeled as matrix partitions.

For each matrix partition problem there is an equivalent trigraph H-colouring problem. We show a 'dichotomy' for the class of list H-colouring problems where H is a so-called trigraph path. For each trigraph path H we show that the list H-colouring problem is either **NP**-complete or polynomial time solvable. For each trigraph path H we associate a digraph H^- such that the list H-colouring problem is polynomial time solvable if the list H^--colouring problem is polynomial time solvable, and is **NP**-complete otherwise.

Keywords: matrix partition problem, digraph homomorphism, list homomorphism, generalized colouring, trigraph, edge-coloured digraph, polymorphism, clique cutset, dichotomy

To Brandy

Acknowledgments

I would like to acknowledge the considerable academic and financial support of my supervisor Dr. P. Hell. As well, I would like to thank Dr. B. Bhattacharya for financial support. The algorithmic approach in Sections 4.3 and 4.4 is based on joint work with Juraj Stacho. The "Graph Homomorphisms Reading Group" has been a huge influence on me; thanks to everyone who came out to play. Finally, I would like to thank my parents and my eternally patient wife Brandy, without whose support and encouragement this thesis would not be possible.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Many problems in graph theory can be formulated in terms of finding a partition of a graph into parts which satisfy some specified conditions. The conditions are often internal conditions on each part, or external conditions between two parts.

For example, consider the familiar k-colouring problem. For a fixed integer k we are required to determine if an input graph G admits a proper k-colouring, i.e. a colouring of the vertices of G with colours from $\{1, ..., k\}$ so that each vertex gets exactly one colour and no edge has endpoints receiving the same colour. We can reformulate the problem in terms of a partition by observing that a graph G admits a proper k-colouring if and only if $V(G)$ admits a partition into k independent sets. In this example we place the *internal* condition that each part be an independent set.

Another well known example is skew partition. Skew partitions have played a prominent role in the study of perfect graphs [12, 13, 16, 15, 18, 53]; [53] gives a historical overview. The *skew partition problem* for an input graph G is defined as follows. Determine if there is a partition of $V(G)$ into four non-empty parts A_1, A_2, B_1, B_2 such that there are no edges between the parts A_1 and A_2, and all possible edges between the parts B_1 and B_2. This is an example of a partition in which *external* conditions are imposed between the parts.

Of course there are examples of partitions which impose both internal and external conditions. The clique-cutset problem is a well known example [28, 35, 55, 59]. Recall that a cutset of a graph is a subset $C \subset V(G)$ such that the graph induced on $V(G) \setminus C$ is disconnected. The *clique-cutset problem* is defined as follows. Given an input graph G

determine if $V(G)$ can be partitioned into three non-empty parts A, B, and C such that there are no edges between parts A and B, and the part C induces a clique. If such a partition exists then C is a clique-cutset of G.

These partition problems share some common features. In each there are a fixed number of parts. Each partition problem has conditions internally—the parts are independent sets, cliques, or have no restriction upon them—and conditions externally—between some pair of parts there are all possible edges, no edges, or no restriction. Feder, Hell, Klein and Motwani in [27, 28] introduced M-partition problems to unify the study of these sorts of partition problems. If we require a partition into m parts, we encode the requirements into an $m \times m$ matrix M with entries from $\{0, 1, \star\}$. An M-partition of a digraph G is a partition of $V(G)$ into parts $V_1, V_2, ..., V_m$ such that the following conditions are met. The entries M_{ij} and M_{ji} encode any external restriction between parts i and j. If $M_{ij} = 0$ then there are no edges from the part V_i to the part V_j, if $M_{ij} = 1$ then there are all possible edges from the part V_i to the part V_j. If $M_{ij} = \star$ there is no restriction. The diagonal entries M_{ii} encode internal restrictions on the part i. If $M_{ii} = 0$ then part V_i is an independent set, if $M_{ii} = 1$ then V_i is a clique. If $M_{ii} = \star$ there is no restriction. By providing a common framework for partition problems, M-partition problems encourage general techniques to be developed which can be applied, not just to some specific partition problem, but to broad classes of partition problems.

Matrix partition problems can be modeled conveniently as trigraph H-colouring problems (cf. for instance [31, 57]). For each matrix partition problem there is an equivalent trigraph homomorphism problem. In fact we can view each matrix M over $\{0, 1, \star\}$ as the adjacency matrix of a trigraph H. Trigraph H-colouring problems are often much more convenient to consider, since we can take advantage of the considerable body of techniques which have been developed to solve digraph homomorphism problems.

Until recently much of the work on partition problems has focused on undirected graphs. Feder, Hell and Tucker-Nally initiated the study of M-partitions of directed graphs in [31]. The authors showed a dichotomy between polynomial and **NP**-complete cases for the list H-colouring problem, for trigraphs H on at most 3 vertices. We continue along these lines, proving a dichotomy for the list H-colouring problem, where H is a so-called trigraph path. This also proves a dichotomy for the list M-partition when M is a tridiagonal matrix.

The organization is roughly as follows. Chapter 1 introduces the M-partition problem and trigraph H-colouring problem, and develops the equivalence between them. Chapter 2

introduces several results which will be useful as tools in later chapters. In Chapter 3 we survey previous results for the M-partition problem. Chapter 4 is devoted to developing a dichotomy for the list H-colouring problem where H is a trigraph path.

1.2 Basic definitions

Complexity Theory

We require a few basic notions from complexity theory. A thorough introduction to these concepts can be found in [34] or [54].

We are primarily interested in *problems*: we are given an input, and required to answer a fixed yes/no question about the input. A problem D is *polynomial time reducible* to a problem D' if there exists a polynomial time algorithm which given an instance d of D produces an instance d' of D', such that d is a "yes" instance of D if and only if d' is a "yes" instance of D'.

We call a problem *polynomial time solvable*, or more informally *polynomial*, if it admits a polynomial time algorithm. The class of all polynomial time solvable problems is denoted by **P**. We call a problem *non-deterministically polynomial time solvable*, if it admits a non-deterministic polynomial time algorithm. The class of all non-deterministically polynomial time solvable problems is denoted **NP**. A problem is called ***NP**-complete* if for each problem in **NP** there exists a polynomial time reduction to our problem.

Each problem in **P** also belongs to **NP**. It is not known if the converse holds; however this is generally not believed to be the case. Thus polynomial problems are often considered to be "easier" than **NP**-complete problems. Much of this thesis is devoted to classifying the computational complexity of a class of trigraph list H-colouring problems. We show that each trigraph list H-colouring problem is polynomial or **NP**-complete, where H is a trigraph path. That is, we show a *dichotomy*, between polynomial and **NP**-complete, for this class of problems. Such dichotomies cannot be assumed *a priori*, since Ladner showed in [48] that if $\mathbf{P} \neq \mathbf{NP}$ then there are infinitely many problems whose complexity is between **P** and **NP**.

Digraphs

A *digraph* G is a set of vertices $V(G)$ along with a binary edge relation $E(G) \subseteq V(G) \times V(G)$. Digraphs will sometimes be called *directed graphs*, or simply *graphs*. A graph is called *symmetric*, or *undirected*, if the edge relation $E(G)$ is symmetric—i.e. $uv \in E(G)$ implies $vu \in E(G)$. The *underlying undirected graph* of a digraph G is the symmetric graph obtained by adding just enough pairs to $E(G)$ to make it a symmetric relation.

Notice that we view symmetric graphs as a special type of directed graphs. This does conflict with the popular usage, however it forces us to explicitly mention when a result applies only to undirected graphs (which is the minority of cases).

For any $u, v \in V$ with $uv \in E(G)$ the ordered pair uv is called an *edge* of G, and we say that the edge is directed from u to v. An edge $uu \in E(G)$ is called a *loop* at u. If $uu \notin E(G)$ then u is called *irreflexive*. A digraph is called *irreflexive* if it has no loops. Sometimes we will require digraphs to be irreflexive, and sometimes we will allow loops. Since we more often require irreflexive digraphs, we assume digraphs are irreflexive and explicitly allow loops when necessary. Sometimes, for clarity, we will write (u, v) for the edge uv. The vertices u and v are said to be *incident* with the edge $uv \in E(G)$. If there is either an edge uv or vu in $E(G)$ we will simply say that u and v are *adjacent*, or that u and v are *neighbours*. Sometimes we will require the following more precise notion of adjacency; if $uv \in E(G)$ then we say that v is an *out-neighbour* of u, and that u is an *in-neighbour* of v. The neighbours of a vertex comprise its *open neighbourhood*. The neighbours of a vertex along with the vertex itself comprise its *closed neighbourhood*. Note that if there is a loop at a vertex then the vertex is in both its open and its closed neighbourhood.

An *induced subgraph* of G is a digraph S defined on a subset of vertices $V(S) \subseteq V(G)$ along with the the edges $E(S) \subseteq E(G)$ which are incident with no vertex outside of $V(S)$. For $X \subseteq V(G)$ the subgraph induced on $V(G) \setminus X$ is denoted $G - X$. If I is an induced subgraph of G which contains no edges then I is called an *independent set* (or *stable set*) of G. If C is an induced subgraph of G which contains all possible edges then C is called a *clique* (or *complete subgraph*) of G.

A *homomorphism* of a digraph G to a digraph H is a mapping $f : V(G) \to V(H)$ such that the following condition holds, for all $u, v \in V(G)$

$$\text{if } uv \in E(G) \text{ then } f(u)f(v) \in E(H).$$

If there is a homomorphism of G to H we denote this fact by the notation $G \to H$. For a

fixed digraph H we define the *H-colouring problem* as follows,

DIGRAPH H-COLOURING
Instance: A digraph G,
Question: Does G admit a homomorphism to the digraph H?

Matrices

Let M be a fixed $m \times m$ matrix with entries $M_{ij} \in \{0, 1, \star\}$; where $i, j = \{1, 2, ..., m\}$ and M_{ij} is the entry in the i^{th} row and j^{th} column of M. We say that M is a matrix *over* the set $\{0, 1, \star\}$. Sometimes we will abbreviate this by saying that M is a 01$\star$-matrix.

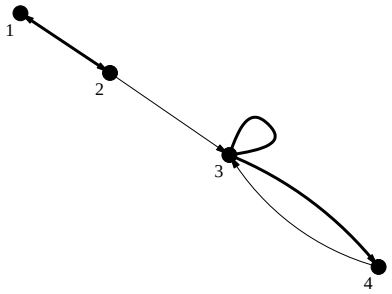
Define the *main diagonal* of M as the set of entries $\{M_{ii}\}$ for all $i = 1, 2, ..., m$. The *sub-diagonal* (resp. *super-diagonal*) is similarly defined as the set of entries $\{M_{(i+1)i}\}$ (resp. $\{M_{i(i+1)}\}$) for $i = 1, 2, ..., (m-1)$. We refer to the sub- and super-diagonals as the *adjacent diagonals* of the main diagonal. A *principal submatrix* M^I is a matrix defined for a subset of indices $I \subseteq \{1, ..., m\}$ as follows. Let M^I be the $|I| \times |I|$ matrix, such that for $i, j \in I$ we have $M^I_{ij} = M_{ij}$.

We now define so-called tridiagonal matrices. Informally speaking, a tridiagonal matrix is one which can have its columns and rows permuted simultaneously so that all non-zero entries appear on the main diagonal and the two adjacent diagonals. More precisely we define a permutation π on the indices of M. Applying π to M we obtain a new matrix $M' = \pi M$ where $M'_{ij} = M_{\pi(i)\pi(j)}$. If there is a permutation π such that, after applying π we obtain a matrix M' such that each non-zero entry appears on the main diagonal and two adjacent diagonals, then M is called a *tridiagonal matrix* and πM is a matrix in *tridiagonal form*. Figure 1.1 shows an example of a matrix in tridiagonal form.

1.3 List matrix partitions

Let M be an $m \times m$ matrix with entries from $\{0, 1, \star\}$. An *M-partition* of a digraph $G = (V(G), E(G))$ is a partition of the vertices $V(G)$ into m parts $\{V_1, V_2, ..., V_m\}$ such that for each $u \in V_i$ and each $v \in V_j$, with $u \neq v$ we have

- if $M_{ij} = 0$ then $uv \notin E(G)$,
- if $M_{ij} = 1$ then $uv \in E(G)$.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 1 & 0 & \star & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & \star & 0 \end{array}\right] \end{array}$$

Figure 1.1: A tridiagonal matrix along with it's associated trigraph. The natural path ordering is indicated for the trigraph.

If $M_{ij} = \star$ then there is no restriction on uv, we can have either $uv \in E(G)$ or $uv \notin E(G)$. Notice that the definition allows empty parts. We can make some simple observations by looking at the matrix. If $M_{ij} = 1$ then there are all possible edges between V_i and V_j; in this case the parts V_i and V_j induce what is sometimes called a *clique-pair*, *skew set*, or *join*. If $M_{ij} = 0$ then there are no possible edges between V_i and V_j; in this case the parts V_i and V_j induce what is sometimes called a *stable-pair*, *disconnected set*, or *co-join*. There is a simple correspondence between a part V_i and the diagonal entry M_{ii}. If $M_{ii} = 0$ then V_i must induce an independent set and if $M_{ii} = 1$ then V_i must induce a clique. (If $M_{ii} = \star$ then V_i is allowed to induce an arbitrary subgraph.) Often we abuse notation and talk about a "part" of the matrix M. In particular, this refers to the part V_i corresponding to some diagonal entry M_{ii} as described above.

For a fixed $01\star$-matrix M, the *M-partition problem* is defined as follows,

M-PARTITION

Instance: A digraph G,

Question: Does G admit an M-partition?

We will mainly be interested in the following "list" version of the M-partition problem. Let $L(G)$ denote a collection of sets $L(v) \subseteq \{1, 2, ..., m\}$, for each $v \in V(G)$. Then a *list M-partition of G with respect to $L(G)$* is an M-partition of G such that for each $v \in V(G)$

we have $v \in V_i$ only if $i \in L(v)$. For a fixed $01\star$-matrix M the *list M-partition problem* is defined as follows,

LIST M-PARTITION
Instance: A digraph G with lists $L(G)$,
Question: Does G admit a list M-partition with respect to $L(G)$?

1.4 Trigraph homomorphism and M-partition

Trigraphs

A *trigraph* H is a set of vertices $V(H)$ with two binary edge relations $E(H)$ and $N(H)$ such that $E(H) \cup N(H) = V(H) \times V(H)$. Notice this means that each pair $ij \in V(H) \times V(H)$ must be in at least one of the two relations $E(H)$ or $N(H)$. We explicitly allow a pair ij to be in both relations, i.e. allow parallel edges between a pair of vertices, so long as they are in different relations. Many of the definitions for digraphs carry over directly to trigraphs, for instance adjacency in the relations $E(H)$ and $N(H)$ has the same definition as adjacency for digraphs. Likewise, we talk about symmetric trigraphs as those with symmetric relations $E(H)$ and $N(H)$. Note that each vertex in a trigraph must have a loop (a pair vv) in at least one of the two relations, and may have loops in both relations. By contrast, recall that the digraphs we consider are not required to have loops.

Trigraph homomorphism

A *homomorphism* of a digraph G to a trigraph H is a mapping $f : V(G) \to V(H)$ such that both of the following conditions hold

- if $uv \in E(G)$ then $f(u)f(v) \in E(H)$ and,
- if $uv \notin E(G)$ and $u \neq v$ then $f(u)f(v) \in N(H)$.

If $f : G \to H$ is a homomorphism of G to H, we say that f *is an H-colouring of G*. There is an alternative, occasionally useful definition of trigraph homomorphism. We can view the non-edges of a digraph as a relation $N(G) = \{uv \notin E(G) \mid u \neq v\}$. Then the above definition of trigraph homomorphism can be restated. A homomorphism of G to H is a

mapping $f : V(G) \to V(H)$ such that $uv \in E(G)$ implies $f(u)f(v) \in E(H)$ and $uv \in N(G)$ implies $f(u)f(v) \in N(H)$.

Let H be a fixed trigraph we can then define the *H-colouring problem* as follows,

TRIGRAPH H-COLOURING
Instance: A digraph G,
Question: Does G admit a homomorphism to the trigraph H?

There is a strong connection between the trigraph H-colouring problem and the digraph H-colouring problem introduced earlier. In fact we can view digraph H-colouring as a special case of trigraph H-colouring. Suppose P and Q are digraphs, and let Q' be the trigraph defined by $E(Q') = E(Q)$ and $N(Q') = V(Q) \times V(Q)$. Then clearly $P \to Q$ if and only if $P \to Q'$ (since *any* mapping $f : V(P) \to V(Q')$ always maps $uv \notin E(P)$ to $f(u)f(v) \in N(Q')$). Thus the class of digraph H-colouring problems we introduced earlier can be viewed as a subclass of the trigraph H-colouring problems just introduced—thus the fact that these problems share the name H-colouring should not cause any ambiguity and we will refer to both as H-colouring problems (without the modifier).

There is also a "list" version of H-colouring. Let H be a trigraph with vertices $V(H) = \{1, 2, ..., m\}$. Let G be a digraph, and let $L(G)$ denote a collection of sets $L(v) \subseteq V(H)$, for each $v \in V(G)$. Then a *list H-colouring of G with respect to $L(G)$* is a homomorphism $f : G \to H$ such that for $v \in V(G)$ we have $f(u) = i$ only if $i \in L(v)$. For a fixed trigraph H, the *list H-colouring problem* is defined as follows,

TRIGRAPH LIST H-COLOURING
Instance: A digraph G with lists $L(G)$,
Question: Does G admit a list homomorphism to the trigraph H with respect to $L(G)$?

Equivalence of trigraph homomorphism and M-partition

List H-colouring and list M-partition describe the same family of problems. We define the *adjacency matrix* of a trigraph H with vertices $V(H) = \{1, ..., m\}$ to be the $m \times m$ matrix M with entries from $\{0, 1, \star\}$ obtained by setting

- $M_{ij} = 0$ if $ij \in N(H) \setminus E(H)$,
- $M_{ij} = 1$ if $ij \in E(H) \setminus N(H)$,
- $M_{ij} = \star$ otherwise (i.e. $M_{ij} = \star$ if $ij \in E(H) \cap N(H)$).

Since $E(H) \cup N(H) = V(H) \times V(H)$ each entry of M is well defined. Clearly, each $01\star$-matrix M is the adjacency matrix of a unique trigraph H. Specifically, if M is an $m \times m$ matrix over $\{0, 1, \star\}$, then H is the trigraph with vertices $V(H) = \{1, ..., m\}$ and relations $E(H) = \{ij \mid M_{ij} = \star \text{ or } 1\}$ and $N(H) = \{ij \mid M_{ij} = \star \text{ or } 0\}$.

Lemma 1.1 *Let M be the adjacency matrix of a trigraph H, and let G be a digraph. Then any M-partition of G induces a homomorphism $G \to H$ and conversely.*

Proof. Let G be a digraph and let H be a trigraph with vertices $\{1, ..., m\}$ and adjacency matrix M.

Let $f : G \to H$ be a homomorphism, then the inverse mapping $f^{-1} : H \to G$ is an M-partition of G. The homomorphism f takes each edge $uv \in E(G)$ to an edge $f(u)f(v) \in E(H)$, and so $M_{f(u)f(v)} \neq 0$. Likewise, f takes each non-edge $uv \notin E(G)$ to an edge $f(u)f(v) \in N(H)$, and so $M_{f(u)f(v)} \neq 1$. Therefore f^{-1} is an M-partition of G.

Now, let $\phi = \{V_1, ..., V_m\}$ be an M-partition of G, then the mapping $h : G \to H$ which takes the each set V_x to x is a homomorphism. Consider an edge $uv \in E(G)$, and suppose $u \in V_x$ and $v \in V_y$, then $M_{xy} \neq 0$ which implies $xy \in E(H)$. Similarly, if we consider a non-edge $uv \notin E(G)$ with $u \in V_x$ and $v \in V_y$, then $M_{xy} \neq 1$; implying that $xy \in N(H)$. Thus h is a homomorphism of G to H. □

Similarly, the list versions of M-partition and trigraph homomorphism are also equivalent.

Some terminology and conventions

We will often refer to *strong* and *weak* edges of a trigraph. The *strong edges* of H are the edges $xy \in E(H) \setminus N(H)$. The *weak edges* of H are the edges $xy \in E(H) \cap N(H)$. (The pairs $xy \in N(H) \setminus E(H)$ are called *non-edges* of H.)

Notice that strong-, weak- and non-edges of H correspond exactly to 1, $\star$ and 0 entries in the adjacency matrix of H. Observe that for any homomorphism $f : G \to H$, the following facts are true for $xy \in V(H) \times V(H)$:

- when xy is a strong edge the sets $f^{-1}(x)$ and $f^{-1}(y)$ must have all possible edges between them,
- when xy is a weak edge the sets $f^{-1}(x)$ and $f^{-1}(y)$ have no restriction on the edges between them, and

- when xy is a non-edge the sets $f^{-1}(x)$ and $f^{-1}(y)$ have no edges between them.

By analogy to digraphs, a trigraph H is called a *trigraph path*, or more simply a *path*, if its adjacency matrix is tridiagonal. Notice, that this is the case precisely when the underlying undirected graph of $E(H)$ is a disjoint union of paths (with loops allowed). A pair of vertices $x, y \in V(H) = \{1, ..., m\}$ are said to be *consecutive* if $x = y \pm 1$. When the adjacency matrix of H is in tridiagonal form, then $xy \in E(H)$ implies that x and y are consecutive. We call an ordering in which adjacent vertices are consecutive a *path ordering* of H. When H is a trigraph path, we will generally assume that $V(H)$ is given in a path ordering and treat the elements of $V(H)$ as integers whenever convenient.

A note about drawing digraphs and trigraphs. Digraphs will be drawn using the usual convention. The vertices are drawn as small circles, and for $uv \in E(H)$ we draw an arrow from u to v. In the case of a symmetric edge we will sometimes choose to omit the arrows and instead draw a single line between u and v to indicate the pair of edges uv and vu. We adopt the following convention when drawing trigraphs. Only the strong and weak edges are shown—the non-edges are implied. Edges are indicated in the usual way, with the strong edges being denoted by thicker lines. Refer to Figure 1.1 or Figure 4.1 for examples. Sometimes two vertices u and v will be joined by a dashed arrow from u to v. In the case of trigraphs, this will emphasize that $uv \in N(H)$. In the case of digraphs, this will mean that there is not an edge between u and v—i.e. $uv \notin E(G)$.

Chapter 2

Tools

2.1 Relational structures and polymorphisms

A *relational structure* H consists of a set of vertices $V(H)$ along with a set of finite arity relations $\{R_1, R_2, ..., R_l\}$ on $V(H)$. We denote the arity of R_i by r_i. The number and arity of the relations determine the *type* of H. Two structures G and H have the *same type* if G and H both have exactly l relations and for $i = 1, ..., l$ the arity of $R_i(G)$ is the same as the arity of $R_i(H)$.

Structures with one binary relation are *digraphs*. Structures with k relations all of which are binary are called *k-edge-coloured digraphs*, or without reference to the number of relations *edge-coloured digraphs*. Notice that the trigraphs introduced earlier are a special kind of 2-edge-coloured digraph, but they *do not* correspond to a type of relational structure because of the additional condition for trigraphs that $E \cup N = V \times V$. Edge-coloured digraphs will be of crucial importance later.

Homomorphisms of relational structures are defined for relational structures of the same type. For relational structures G and H of the same type, a mapping $f : V(G) \to V(H)$ is a *homomorphism* of G to H if for all $i = 1, ..., l$ we have $(x_1, x_2, ..., x_{r_i}) \in R_i(G) \Rightarrow (f(x_1), f(x_2), ..., f(x_{r_i})) \in R_i(H)$. We say a mapping $f : V(G) \to V(H)$ is a *homomorphism with respect to a relation* R_i if it satisfies the homomorphism condition with respect to that R_i—i.e. we have that $(x_1, ..., x_{r_i}) \in R_i(G)$ implies $(f(x_1, ..., f(x_k)) \in R_i(H)$.

The *constraint satisfaction problem* for H is denoted CSP(H) and is defined as follows. Given an input structure G of the same type as H, determine if G admits a homomorphism

to H. We say that H is a *conservative structure* if H contains all possible unary relations—i.e. for every subset $X \subseteq V(H)$ there is a corresponding unary relation $R_X(H) = \{x \in X\}$. If H is a conservative structure, then CSP(H) is called a *conservative constraint satisfaction problem*, and denoted c-CSP(H). Given an instance G of c-CSP(H) a vertex $v \in R_X(G)$ can only map to a vertex in $R_X(H) = X \subseteq V(H)$. Thus the presence of all possible unary relations on $V(H)$ is equivalent to having arbitrary lists of allowable images for the vertices of G. This problem has also been called the *list constraint satisfaction problem*. It should then be clear that the list H-colouring problem for a digraph H is equivalent to c-CSP(H') where H' is the conservative structure obtained from H by adding all $2^{|V(H)|}$ possible unary relations.

Let H be a fixed relational structure, and k a fixed positive integer. A *k-ary operator* on $V(H)$ is a mapping of $V(H)^k \to V(H)$. An operator is called *conservative* if it always returns one of its arguments. For example the logical disjunction and conjunction operators $\vee$ and $\wedge$ are conservative, but the negation operator $\neg$ is not.

For relational structures G and H of the same type the *direct product* $G \times H$ is defined as follows. The structure $G \times H$ has the same type as G and H, with vertices $V(G \times H) = V(G) \times V(H)$ and for relation R_i of arity r_i, define $R_i(G \times H) = \{(x_1y_1, ..., x_{r_i}y_{r_i}) \mid (x_1, ..., x_{r_i}) \in R_i(G) \text{ and } (y_1, ..., y_{r_i}) \in R_i(H)\}$. The *$k^{th}$ direct power* of a structure is defined in the obvious way as the k-fold product $H^k = H \times H \times \cdots \times H$ of a structure H with itself.

A *k-ary polymorphism* φ of a relation $R_i(H)$ is a k-ary operator on $V(H)$ such that the mapping $\varphi : H^k \to H$ is a homomorphism with respect to $R_i(H)$. A *k-ary polymorphism* φ of a relational structure H is a k-ary operator on $V(H)$ such that the mapping $\varphi : H^k \to H$ is a homomorphism. For example, when H is a digraph, to verify that a ternary operator φ is a polymorphism of H we check for all triples of edges $xx', yy', zz' \in E(H)$ that $\varphi(x, y, z)\varphi(x', y', z')$ is also an edge of $E(H)$. It is worth noting that the polymorphisms of a conservative structure are necessarily conservative since they must preserve all unary relations. For example, suppose H is a conservative structure with an k-ary polymorphism φ. For $x_1, x_2, ..., x_k \in V(H)$ the value of $\varphi(x_1, x_2, ..., x_k)$ must be an element in the set $X = \{x_1, x_2, ..., x_i\}$, since $R_X(H) = X$ is a unary relation of H which is preserved by φ.

There is a special type of polymorphism which all relational structures have. A *projection* is an operation which ignores its input, except for some specified co-ordinate position. For example, if for some fixed $i \in \{1, ..., k\}$ we have $\varphi(x_1, x_2, ..., x_k) = x_i$, for all k-tuples

$(x_1, x_2, ..., x_k) \in V(H^k)$, then φ is a *projection onto the i^{th} coordinate*. It is easy to see from the definition of direct products that each projection is a polymorphism.

Certain types of polymorphism can be used to obtain polynomial algorithms for CSP(H). There are three important types of polymorphisms which are well-known to imply polynomial algorithms for CSP(H)—these polymorphism types are particularly important for conservative structures.

An operator f is a *majority operator* if f is ternary and for all $x, y \in V(H)$ we have $f(x,x,y) = f(x,y,x) = f(y,x,x) = x$. It is well-known that CSP(H) is solvable in polynomial time if H admits a polymorphism which is a majority operator (cf. for instance [33, 45]).

An operator f is a *semi-lattice operator*, if f is a binary operator which is commutative, associative, and idempotent. Each semi-lattice operator f corresponds to a partial order $\leq_f$ on $V(H)$, where $f(a,b) = f(b,a) = a$ if and only if $a \leq_f b$. Note that we could have $f(a,b) = f(b,a) = c$ for some $c \notin \{a,b\}$ in which case a and b are incomparable in the partial order $\leq_f$. Polymorphism which are semi-lattice operators are also known to lead to polynomial algorithms for CSP(H) and have been extensively studied [36, 41, 42, 45, 46].

When H is a conservative structure, then any semi-lattice polymorphism f is also conservative. This implies each a and b in $V(H)$ are comparable in $\leq_f$, and thus in this case f corresponds to a total order on $V(H)$. An *$\underline{X}$-enumeration* of a digraph H, is a total order $<_H$ on $V(H)$, where $min(a,b) = min(b,a) = a$ if and only if $a <_H b$, such that for any two edges xy, $x'y' \in E(H)$ we have the minimum $min(x,x')min(y,y') \in E(H)$. An $\underline{X}$-enumeration of a digraph H can be used to solve the list H-colouring problem in polynomial time [36]. In fact the function $min(\_,\_)$ is a conservative semi-lattice polymorphism, since verifying that $min(x,x')min(y,y')$ is an edge when $xy, x'y'$ are edges is the same as verifying that $min : H^2 \to H$ is a homomorphism. Thus, a digraph H admits a conservative semi-lattice polymorphism exactly when it admits a $\underline{X}$-enumeration. In a separate line of research, relational structures which admit conservative semi-lattice polymorphisms have also been studied, they are called *max-closed* in [45, 46]. Jeavons and Cohen showed in [46] that if a relational structure H is max-closed then CSP(H) is polynomial time solvable.

An operator f is a *Mal'tsev operator* if f is ternary and for all $x, y \in V(H)$ we have $f(x,x,y) = f(y,x,x) = y$. It has recently been shown that Mal'tsev polymorphisms also lead to polynomial algorithms for $CSP(H)$ [6, 5].

Suppose f is a conservative polymorphism of H, and B is a subset of $V(H)$. Then

we say f is a semi-lattice operator on the subset B if the restriction $f|_B : B^2 \to B$ is a semi-lattice operator. We likewise say f is a majority (or Mal'tsev) operator on B if the restriction $f|_B : B^3 \to B$ is a majority (or Mal'tsev) operator.

When H is a conservative structure Bulatov was able to show in [7], that if for every two-element subset B there is a polymorphism f of H such that the restriction $f|_B$ is semi-lattice, majority, or Mal'tsev, then the conservative constraint satisfaction problem c-CSP(H) is polynomial; otherwise c-CSP(H) is **NP**-complete.

Proposition 2.1 ([7]) *Let H be a conservative relational structure. Then c-CSP(H) is solvable in polynomial time if for each two-element subset $B \subseteq V(H)$, there exists a polymorphism f of H such that the restriction $f|_B$ is a* semi-lattice, *a* majority *or a* Mal'tsev *operator on the subset B. Otherwise c-CSP(H) is **NP**-complete.*

This result will be of crucial importance in Chapter 4, where we will apply it to show that certain classes of edge-coloured digraphs admit polynomial list H-colouring algorithms.

Proposition 2.1 tells us that for every two-element subset B there is some polymorphism which is a semi-lattice, majority or Mal'tsev operator on B. The following Proposition tells us that if c-CSP(H) is polynomial there will always be a set of three polymorphisms which satisfy the conditions of Proposition 2.1.

Proposition 2.2 ([7]) *Let H be a conservative relational structure, for which c-CSP(H) is polynomial. Then there exist polymorphisms f,g,h such that for every two-element subset $B \subseteq V(H)$ either $f|_B$ is a* semi-lattice *operator, $g|_B$ is a* majority *operator, or $h|_B$ is a* Mal'tsev *operator.*

We mentioned above that trigraphs are a special type of 2-edge-coloured digraph. It is also possible to view digraphs as 2-edge-coloured digraphs in the following way. Let $G = (V(G), E(G))$ be a digraph, then we can "add" a relation $N(G) = \{uv \notin E(G) \mid u \neq v\}$. Referring to the definition of trigraph homomorphism in Section 1.4 it is easy to see that there is a list H-colouring of G (where G is a digraph and H is a trigraph) if and only if there is a homomorphism from G to H (taken as 2-edge-coloured digraphs). We can thus view instances of trigraph list H-colouring as instances of c-CSP(H). However, not all instances of c-CSP(H) correspond to instances of list H-colouring. Thus Proposition 2.1 cannot be applied directly to classify the complexity of trigraph list H-colouring. For example, a polynomial c-CSP(H) problem implies a polynomial list H-colouring problem. However,

we can demonstrate problems which are **NP**-complete according to Proposition 2.1 when regarded as c-CSP problems but become polynomial, or quasi-polynomial, when regarded as trigraph H-colouring problems. Appendix A provides examples of such behaviour.

2.2 "Small" relations and majority polymorphisms

We will now prove a useful fact about "small" relations. We will show that if we have a relation R which is a subset of $L \times L'$, for two sets L, L' of size at most 2, then any majority operator is a polymorphism of R.

Lemma 2.3 *Let X be a set and let L and L' be subsets of X such that $|L| \leq 2$ and $|L'| \leq 2$. If g is a majority operator on X, then g is a polymorphism of any relation $R \subseteq L \times L'$.*

Proof. Let D be the digraph with vertex set $V(D) = X$, and edge relation $R(D) = R$. Let g be a majority operator. We verify that $g : D^3 \to D$ is a homomorphism. Let (w_1, w_2, w_3) be a vertex of D^3. Observe that if w_1, w_2, w_3 are all distinct then at most two of them are in L, and at most two in L'. In this case the vertex (w_1, w_2, w_3) is isolated in D^3, since all edges of $R(D)$ are directed from L to L'. Thus, it suffices to consider the value of g on triples which are not all distinct. Suppose that $(x_1, x_2, x_3)(y_1, y_2, y_3)$ is an edge of R^3, where x_1, x_2, x_3 are not all distinct, and y_1, y_2, y_3 are not all distinct. Then, since g is a majority operator, $g(x_1, x_2, x_3)g(y_1, y_2, y_3)$ is one of the edges x_1y_1, x_2y_2 or x_3y_3 and so g is a polymorphism. □

2.3 Choosing representatives

Let H be a fixed trigraph. The *list H-colouring problem with representatives* is a restriction of the general list H-colouring problem in which the input is restricted so that only certain types of lists $L(G)$ are allowed.

LIST H-COLOURING WITH REPRESENTATIVES

Instance: A digraph G, lists $L(G)$, a pair of sets $S \subseteq V(H)$ and $V_S \subseteq V(G)$ such that,

- *for each $v \in V(G)$ the list $L(v) \subseteq S$, and*
- *for each $x \in S$ there is a fixed $v_x \in V_S$ such that $L(v_x) = \{x\}$;*

Question: Does G admit a list H-colouring with respect to $L(G)$?

There are two notable features of the list H-colouring with representatives. Firstly, for

each $x \in S$ we have *pre-coloured* one vertex v_x by requiring that $L(v_x) = \{x\}$. We call the vertex v_x a *representative* of x. A *choice of representatives* for S is a set $V_S \subseteq V(G)$ such that for each $x \in S$ there is exactly one $v_x \in V_S$. The second notable feature is that each $L(v)$ is a subset of S, forcing the parts not in S to be *empty*. These parts appear on no list in $L(G)$ and can be safely ignored; no vertex will map there in any list H-colouring. That is, each solution to the list $(H - S)$-colouring problem is a solution to the list H-colouring problem. When we consider instances of the list H-colouring problem with representatives in later chapters we will without loss of generality make the simplifying assumption that each part has a representative.

Introducing representatives does not change the essential nature of the list H-colouring problem, as shown in the following lemma.

Lemma 2.4 *Let H be a trigraph. The list H-colouring problem with representatives is solvable in polynomial time if and only if the list H-colouring problem is solvable in polynomial time.*

Proof. An instance of list H-colouring with representatives is an instance of list H-colouring. We will demonstrate how to transform an instance of the list H-colouring problem into a polynomial number of instances of the list H-colouring problem with representatives.

Let H be a trigraph with k vertices and let G be a digraph on n vertices with lists $L(G)$ be an instance of the list H-colouring problem. We construct an instance of the list H-colouring problem for each possible choice of a set S and representatives V_S by modifying lists appropriately. Since the size of H is fixed, there are constantly many choices for a set S. For each choice of S there are $O(n^{|S|})$ choices of representatives V_S. Thus the total number of instances is bounded by a polynomial in the size of the input. The original list H-colouring problem has a solution if and only if at least one of these list H-colouring problems with representatives has a solution. □

When we have an instance of the list H-colouring problem, we can often use the fact that we have representatives to simplify the lists.

2.4 Consistency checks for lists

Given a graph G with lists $L(G)$, an instance of a list H-colouring problem, it is often possible to reduce lists in the following way. We say that *a set of lists $L(G)$ can be reduced*

to lists $L^*(G)$ if both of the following conditions are met:

- for each $v \in V(G)$ we have $L^*(v) \subseteq L(v)$, and
- there is a homomorphism $G \to H$ with respect to reduced lists $L^*(G)$ if and only if there is a homomorphism $G \to H$ with respect to the original lists $L(G)$.

That is, $L^*(G)$ is a set of *reduced lists* for $L(G)$ if $L^*(G)$ can be obtained from the lists in $L(G)$ by removing elements which are not consistent with any homomorphism of $G \to H$.

The following *arc-consistency procedure* is a simple and well-known way of obtaining reduced lists. It has been introduced in several contexts, and has enjoyed enduring popularity (cf. for instance [19, 36, 40, 42, 45]). The arc-consistency procedure can be applied to any relational structure. Given a set of lists $L(G)$ the procedure produces reduced lists $L^*(G)$ satisfying the following condition. For each relation R, and each two-element subset $\{u, v\}$ of $V(G)$, if we have x in the list $L^*(u)$ then there exists y in the list $L^*(v)$ such that the mapping $u \mapsto x, v \mapsto y$ is a homomorphism with respect to R. We will outline the procedure for edge-coloured digraphs.

Procedure 2.5 (Arc-Consistency for edge-coloured digraphs)
Given edge coloured digraphs G and H of the same type and list $L(G)$. Initially let $L^*(G) = L(G)$. As long as changes occur, for all relations R and each pair $uv \in R(G)$ update $L^*(u)$ and $L^*(v)$ as follows:

1. *Remove from $L^*(u)$ any x for which there is no $y \in L^*(v)$ such that $xy \in R(H)$,*
2. *Remove from $L^*(v)$ any y for which there is no $x \in L^*(v)$ such that $xy \in R(H)$.*

Return the reduced lists $L^*(G)$.

If each list in $L^*(G)$ is non-empty, we say that *arc-consistency succeeded*; otherwise if some list $L^*(v)$ is empty we say that *arc-consistency failed*. Arc-consistency is a very useful procedure. In fact, when H has a conservative semi-lattice polymorphism, the fact that arc-consistency succeeded for all relations, on input G and $L(G)$, is sufficient to guarantee the existence of a homomorphism $G \to H$ with respect to $L(G)$ [36, 40, 42, 45].

When considering the trigraph list H-colouring problem a similar arc-consistency procedure can be applied.

Procedure 2.6 (Arc-consistency for trigraphs)
Given a trigraph H, digraph G and lists $L(G)$. Initially let $L^*(G) = L(G)$ and as long as changes occur for $uv \in V(G) \times V(G)$:

1. *if $uv \in E(G)$:*
   - *Remove from $L^*(u)$ any x for which there is no $y \in L^*(v)$ such that $xy \in E(H)$*
   - *Remove from $L^*(v)$ any y for which there is no $x \in L^*(v)$ such that $xy \in E(H)$*
2. *if $uv \notin E(G)$ and $u \neq v$:*
   - *Remove from $L^*(u)$ any x for which there is no $y \in L^*(v)$ such that $xy \in N(H)$*
   - *Remove from $L^*(v)$ any y for which there is no $x \in L^*(v)$ such that $xy \in N(H)$*

Return the reduced lists $L^*(G)$.

Note that, arc-consistency ensures that if x is a strong loop then for all $u \in V(G)$ we have $x \in L^*(u)$ only if u is a symmetric neighbour of v_x; and if xy is a strong edge then for all $u \in V(G)$ we have $y \in L^*(u)$ only if $v_x u \in E(G)$.

Both of the consistency procedures introduced above are guaranteed to produce reduced lists $L^*(G)$, since they remove from lists in $L(G)$ only elements which are inconsistent with any homomorphism.

2.5 Instances of list H-colouring with lists of size at most 2

When the list H-colouring problem is restricted to instances with lists of size at most two, then there is a polynomial algorithm for list H-colouring.

Lemma 2.7 *[28, 31, 33] Let H be a trigraph. Then the list H-colouring problem, restricted to instances consisting of a digraph G and lists $L(G)$ such that each list in $L(G)$ has size at most 2, is polynomial time solvable.*

Proof. The standard proof of this fact is by reduction to the well-known **2SAT** problem, for which there are efficient algorithms known (cf. for instance [1, 20]). There is a different approach which demonstrates the usefulness of Lemma 2.3 and will be helpful in understanding a similar construction in Chapter 4.

We first make a general observation which applies to any instance of the list H-colouring problem. Suppose uv is an ordered a pair of distinct vertices from G and let $f : G \to H$ be a homomorphism with respect to $L(G)$. Then the restriction $f|_{\{u,v\}}$ maps the pair uv into either the relation $(L(u) \times L(v)) \cap E(H)$, in the case that $uv \in E(G)$, or the relation $(L(u) \times L(v)) \cap N(H)$, in the case that $uv \notin E(G)$.

We can apply this observation to prove Lemma 2.7 by replacing $E(H)$ and $N(H)$ with these binary relations derived from lists. Our approach is the following. We construct two edge-coloured digraphs of the same type; H^* on $V(H)$, and G^* on $V(G)$. For each pair of (not necessarily distinct) subsets L and L' from $V(H)$ we construct the following relations. Add a relation $E^{L,L'}(H^*) = (L \times L') \cap E(H)$ to H^* and a relation $E^{L,L'}(G^*) = \{uv \in E(G) \mid L(u) \subseteq L, L(v) \subseteq L'\}$ to G^*. Likewise add a relation $N^{L,L'}(H^*) = (L \times L') \cap N(H)$ to H^* and a relation $N^{L,L'}(G^*) = \{uv \notin E(G) \mid L(u) \subseteq L, L(v) \subseteq L', u \neq v\}$ to G^*. It is straightforward to check that $G \to H$ with respect to $L(G)$ exactly when $G^* \to H^*$ with respect to $L(G)$.

If each list in $L(G)$ has size at most 2, then each relation in H^* is a subset of a product of two sets of size at most two. Thus, in this case, H^* has a conservative majority polymorphism by Lemma 2.3. Thus list H^*-colouring is polynomial implying that list H-colouring is also polynomial (in the restricted case that all lists have size at most 2). This completes the proof of Lemma 2.7. □

2.6 Sparse-dense partitions of digraphs

A class of graphs $\mathcal{C}$ is closed under taking induced subgraphs if $G \in \mathcal{C}$ implies all induced subgraphs of G are also in $\mathcal{C}$. Suppose we have two classes of graphs $\mathcal{A}$ and $\mathcal{B}$ both of which are closed under taking induced subgraphs. And further, suppose there is a constant c such that any $G \in \mathcal{A} \cap \mathcal{B}$ has size $|G| \leq c$. For two classes $\mathcal{A}$ and $\mathcal{B}$, meeting the above conditions, we call $\mathcal{A}$ and $\mathcal{B}$ *sparse-dense* with respect to one another. For example when $\mathcal{A}$ is the class of all independent sets and $\mathcal{B}$ is the class of all cliques, these two classes are sparse-dense with respect to one another.

Then *sparse-dense partition* of a digraph G is a partition of $V(G)$ into two sets inducing digraphs $G_A \in \mathcal{A}$ and $G_B \in \mathcal{B}$, where $\mathcal{A}$ and $\mathcal{B}$ are classes of graphs which are sparse-dense with respect to one another. If membership in $\mathcal{A}$ and $\mathcal{B}$ can be tested in polynomial time, then all possible sparse-dense partitions of G can be enumerated in polynomial time.

Proposition 2.8 *([28]) Let $\mathcal{A}$ and $\mathcal{B}$ be two classes of graphs which are sparse-dense with respect to one another. Then any digraph G on n vertices has at most n^c sparse-dense partitions into $G_A \in \mathcal{A}$ and $G_B \in \mathcal{B}$. Further, if membership in each of $\mathcal{A}$ and $\mathcal{B}$ is testable in polynomial time, then all sparse-dense partitions can be enumerated in polynomial time.*

Sparse-dense partitions are a particularly useful tool for solving M-partition problems when M is a matrix with diagonal entries over $\{0, 1\}$. We permute rows and columns of M so that all zeros on the main diagonal appear before any one. We then partition M into diagonal blocks A (containing only diagonal 0's) and B (containing only diagonal 1's). Let C and D be the off-diagonal block matrices corresponding to the remaining entries of M.

To use Proposition 2.8, we verify that the classes of A-partitionable graphs and B-partitionable graphs are sparse-dense with respect to each other. Firstly, observe that, for any matrix M, the class of M-partitionable graphs is always closed under taking induced subgraphs. Now consider a graph G which is which is both A- and B-partitionable. Then, since G is B-partitionable, G can be covered by l cliques. Each of the l cliques contains at most k vertices, otherwise G is not A-partitionable. And so, $V(G)$ has at most $k \times l$ vertices. Thus the classes of A- and B-partitionable graphs are sparse-dense with respect to one another. Proposition 2.8 tells us that if there is a polynomial algorithm for both A-partition and B-partition, then all possible sparse-dense partitions of an input graph can be enumerated in polynomial time.

Consider the following example. Let M be a matrix over $\{0, 1\}$ and let A be the submatrix of M with only zeros on the diagonal, and B be the submatrix of M with only ones on the main diagonal. The A- and B-partition problems are easily seen to be polynomial. Indeed, consider an A-partition of a digraph G. Observe that two vertices of $u, v \in V(G)$ can be placed in a part V_i of A if and only if u and v have the same open in- and out-neighbourhoods. Thus, we can partition the input digraph into equivalence classes based on open neighbourhoods, at most $|A|$ classes can be partitioned by A so, in polynomial time, we can try all possible assignments of equivalence classes to parts of A. Likewise, considering a B-partition of G observe that two vertices $u', v' \in V(G)$ can be placed in a part V_j of B if and only if they have the same closed in- and out-neighbourhoods; and we have a polynomial algorithm for B partition. As observed above, the classes of A and B partitionable graphs are sparse-dense with respect to one another. Thus, we may assume

that we have a sparse-dense partition of our input graph G into (G_A, G_B). With a sparse-dense partition in hand finding an M-partition is easy. Thus we can partition G_A and G_B into equivalence classes based on neighbourhoods. Since the size of M is fixed, we can try all possible assignments of equivalence classes to parts of M in polynomial time.

Chapter 3

Previous Results

In this chapter we survey existing results for matrix partition problems. These can equivalently be seen as results for trigraph homomorphism problems. We will for the most part state these results in terms of matrices but the translations are straightforward.

The layout of this chapter is as follows. Firstly we will present in Section 3.1 some basic facts about M-partitions which will help in interpreting the results that follow. We then survey in Section 3.2 what is known about symmetric matrices M. Finally, in Section 3.3 we summarize what is known in the more general case of M-partitions where M is not necessarily symmetric.

3.1 Basic facts about M-partitions

The *complement* of a digraph G is the digraph $\overline{G}$ on $V(G)$ with edges $E(\overline{G}) = \{uv \mid u \neq v \text{ and } uv \notin E(G)\}$. The *complement* of an $m \times m$ matrix M is the matrix $\overline{M}$ obtained from M by interchanging 0 and 1—i.e. if $M_{ij} = 0$ then $\overline{M}_{ij} = 1$ (the $\star$ entries of M and $\overline{M}$ are identical). The following easy proposition tells us that the M-partition and $\overline{M}$-partition problems are the same.

Proposition 3.1 *Let M be a matrix over $\{0, 1, \star\}$, let G be a digraph, and let ϕ be an M-partition of G. Then ϕ is also an $\overline{M}$-partition of $\overline{G}$.*

This is easy to see, since the definition of an M-partition of G is symmetric—i.e. we only need to interchange 0 and 1 in the definition of M-partition to observe that ϕ is in fact an $\overline{M}$-partition of $\overline{G}$.

Recall that the adjacency matrix of a digraph H is the matrix M over $\{0,1\}$, where $M_{ij} = 1$ if and only if $ij \in E(H)$. (Notice that we could have $i = j$ and so loops are not precluded by this definition.)

Proposition 3.2 *Let H be a digraph with loops allowed and with adjacency matrix M. Let M' be the matrix obtained from M by replacing each 1 by a $\star$. Then each H-colouring of a digraph G is also an M'-partition of G, and conversely.*

This is best seen by considering the relationship between digraph homomorphisms and trigraph homomorphisms (see Section 1.4). Construct a trigraph H' on $V(H)$ with edges $E(H') = E(H)$ and $N(H') = V(H) \times V(H)$. Clearly a digraph G is H-colourable if and only if it is also H' colourable—*any* mapping of $V(G)$ to $V(H)$ maps each non-edge of G to an edge in $N(H')$. Since H and H' share the same E relation, the adjacency matrices M and M' will be identical—except that M' will have a $\star$ wherever M has a 1.

Proposition 3.2 is quite useful, since any time we get a complexity classification of a class of digraphs we get an equivalent classification for the class of corresponding matrices. As an example consider the M-partition problem without lists where M is a symmetric matrix over $\{0,\star\}$ with all 0 on the main diagonal. We can easily obtain a classification of such matrices into **NP**-complete and polynomial by the considering the corresponding graph homomorphism problems. By Proposition 3.2 each matrix M corresponds to some digraph H which is symmetric and has no loops. A well know result of Hell and Nešetřil [39] classifies the complexity of the H-colouring problem (without lists) for irreflexive symmetric graphs as polynomial if H is a bipartite graph, and **NP**-complete otherwise. We thus get our classification: the M-partition problem is polynomial if M corresponds to a bipartite graph H, and is **NP**-complete otherwise.

When considering M-partition problems with lists, there is a simple observation which drastically simplifies the classification project. The complexity of list M-partition problems is monotone, in the sense that if a matrix M has an **NP**-complete list partition problem then any matrix M' which contains M as a principal submatrix also has an **NP**-complete list partition problem.

Proposition 3.3 *If M' is a fixed matrix with principal submatrix M such that the list M-partition problem is **NP**-complete, then the list M'-partition problem is also **NP**-complete.*

We can easily reduce the list M-partition problem to the list M'-partition problem since M is a principal submatrix of M'. Given an instance of the list M-partition problem, a digraph G with lists $L(G)$, we define an instance of the list M'-partition problem with the same digraph G and the same lists $L(G)$.

An algorithm is called *quasi-polynomial*, in [28], if its running time is bounded by $n^{O(logn)}$, where n is the size of the input. In terms of M-partition problems, we let n be the number of vertices in an input instance. In [22] the authors were able to prove the following general result about all list M-partitions.

Proposition 3.4 ([22]) *Let M be any matrix over $\{0, 1, \star\}$ then the list M-partition problem is **NP**-complete or quasi-polynomial.*

Since each list M-partition problem is **NP**-complete or quasi-polynomial the above result has been termed a *quasi-dichotomy* in [28]. For some classes of matrices this has been strengthened to a dichotomy between **NP**-complete or polynomial. For example, all matrices over $\{0, 1\}$, $\{0, \star\}$ and $\{1, \star\}$ have **NP**-complete or polynomial list M-partition problems—we describe these results in detail later.

3.2 Partitions of undirected graphs

3.2.1 Small symmetric matrices

The list matrix partition problem is polynomial time solvable when M is a fixed matrix of size at most 2×2—since in this case all lists can have size at most two and is thus polynomial by Lemma 2.7.

Proposition 3.5 ([28]) *Let M be a fixed $m \times m$ matrix over $\{0, 1, \star\}$, where $m \leq 2$, then the list M-partition problem is solvable in polynomial time.*

When M is symmetric with size $m = 3$ there are **NP**-complete cases. Probably the most familiar is *graph 3-colouring*. In the graph 3-colouring problem we are given an input graph G and required to determine if G is partitionable into 3 independent sets. The other well-known problem is *stable cutset*. Given an input graph G we need to find a (non-empty) independent set whose removal separates the remaining vertices in $V(G)$ into two (non-empty) non-adjacent parts. Figure 3.1 shows the corresponding matrices. The M-partition problem for both of these matrices is known to be **NP**-complete even without lists [17, 47].

$$\begin{bmatrix} 0 & \star & \star \\ \star & 0 & \star \\ \star & \star & 0 \end{bmatrix} \qquad \begin{bmatrix} \star & \star & 0 \\ \star & 0 & \star \\ 0 & \star & \star \end{bmatrix}$$

Figure 3.1: The matrices for 3-colouring (left) and stable cutset (right). Both matrices (and their complements) have **NP**-complete list M-partition problems.

In [28, 27] Feder, Hell, Klein and Motwani show that for all other symmetric matrices M with size $m = 3$ the list M-partition problem is polynomial time solvable.

Proposition 3.6 ([28]) *Let M be a fixed symmetric matrix over $\{0, 1, \star\}$, with size $m \leq 3$. If M is the matrix for 3-colouring, or stable cutset (or their complements) then the list M-partition problem is **NP**-complete; otherwise the list M-partition problem is polynomial time solvable.*

That is, matrices of size at most 3 have a *dichotomy* property. Each list M-partition problem with size $m \leq 3$ can be classified as either **NP**-complete or polynomial time solvable. Feder, et al. [28] were also able to show a dichotomy for the complexity of the M-partition problem (without lists) for symmetric matrices M with at most 4 parts.

Proposition 3.7 ([28]) *Let M be a symmetric matrix, with size $m \leq 4$. If M contains the matrix for 3-colouring (or its complement) as a principal submatrix, and there is no $\star$ on the main diagonal of M then the M-partition problem (without lists) is **NP**-complete; otherwise the M-partition problem (without lists) is polynomial time solvable.*

Notice that a $\star$ on the main diagonal of M makes finding an M-partition (without lists) trivial, since we can simply map the entire input graph to the part corresponding to the diagonal $\star$. However, applications often require that all parts be non-empty. For example recall the clique cutset problem, we need three non-empty sets A, B and C, such that C is a clique and there are no edges between A and B. If we allow some part to be empty then the set C may not be a cutset. To deal with this trivial difficulty so-called surjective M-partitions were introduced. A *surjective M-partition problem* is an M-partition problem (without lists) with the additional constraint that each part be non-empty. Surjective M-partitions lie somewhere between M-partitions and list M-partitions in complexity. An M-partition problem can be reduced to several instances of surjective partition, by considering the surjective partition problems for the set of all principal submatrices of M. If an input

graph G admits an M-partition then G also admits a surjective partition by one of these submatrices. It is trivial to reduce the surjective M-partition problem to the list M-partition problem with representatives. Initially we set all lists to $\{1, ..., m\}$, then consider all possible choices of representatives in which no part is empty.

Dantas, de Figueiredo, Gravier and Klein have studied the surjective M-partition problem for small symmetric matrices [14](cf. also [16]). In [14] the authors show that—with one exception—all 4×4 matrices with $\star$-diagonals admit polynomial algorithms for M-partition with all parts non-empty.

Proposition 3.8 ([14]) *Let M be a fixed symmetric matrix over $\{0, 1, \star\}$, with size $m = 4$ and all $\star$'s on the main diagonal, such that M is not the matrix $2K_2$—shown in Figure 3.2—(or its complement). Then surjective M-partition is solvable in polynomial time.*

The exceptional matrix with open complexity for surjective M-partition is $2K_2$ (shown in Figure 3.2). This is the complement of a $0\star$-matrix and so corresponds to a digraph homomorphism problem. The list version of this problem is shown **NP**-complete in [21], but the surjective case eludes classification.

The list M-partition problem for matrices of size $m \leq 4$ has also been studied. In [10] Cameron, Eschen, Hoàng and Sritharan, expanding on results from [28] and [18], showed (with one exception) that list M-partitions with $m = 4$ have a dichotomy. The one exception is the so-called *stubborn problem*(and its complement) shown in Figure 3.3. A matrix M which contains one of the **NP**-complete 3×3 matrices—shown in Figure 3.1—(or their complements) obviously has an **NP**-complete list M-partition problem. In addition Cameron, et al. identified two 4×4 matrices—shown in Figure 3.2—(and their complements) which have **NP**-complete list M-partition problems. The first is $2K_2$ *partition* which was shown **NP**-complete in [21], we are given a graph G and required to determine if G is list partitionable into 4 parts A_1, A_2, B_1, B_2 such that there are all possible edges between A_1 and A_2, and all possible edges between B_1 and B_2. The second is *stable cutset pair partition* shown **NP**-complete in [25](cf. also Proposition 3.15), given a graph G we are required to determine if G is list partitionable into 4 parts A, B_1, B_2, C, such that B_1 and B_2 are independent sets and there are no edges between $A \cup B_1$ and C, and no edges between A and $B_2 \cup C$. Cameron, et al. demonstrate polynomial algorithms for all 4×4 matrices, except for these 2 matrices and the stubborn problem (and their complements).

$$\begin{bmatrix} \star & 1 & \star & \star \\ 1 & \star & \star & \star \\ \star & \star & \star & 1 \\ \star & \star & 1 & \star \end{bmatrix} \qquad \begin{bmatrix} \star & \star & 0 & 0 \\ \star & 0 & \star & 0 \\ 0 & \star & 0 & \star \\ 0 & 0 & \star & \star \end{bmatrix}$$

Figure 3.2: Two matrices of order 4 with **NP**-complete list partition problems. On the left is the matrix $2K_2$—in complemented form this is known as the *Winkler partition* problem. The $\overline{M}$-partition problem for this matrix corresponds to the list H-colouring problem where H is the digraph C_4^o. On the right is the matrix for *stable cutset pair*.

$$\begin{bmatrix} 0 & \star & 0 & \star \\ \star & 0 & \star & \star \\ 0 & \star & \star & \star \\ \star & \star & \star & 1 \end{bmatrix}$$

Figure 3.3: The matrix for the so-called stubborn problem. The complexity of the list M-partition problem for this matrix remains open.

Proposition 3.9 ([10, 11]) *Let M be a fixed symmetric matrix over $\{0, 1, \star\}$, with size $m = 4$, such that M is not the stubborn matrix (or its complement)—shown in Figure 3.3. Then if M contains the matrix for 3-colouring, stable cutset, stable cutset pair, or $2K_2$—shown in Figures 3.1 and 3.2—(or their complements) the list M-partition problem is **NP**-complete; otherwise the list M-partition problem is polynomial time solvable.*

The exceptional case, the stubborn problem, is shown in [22, 28] to admit a *quasi-polynomial* algorithm. In [30] the authors improved the bound to an $n^{O(log(n)/log\,log(n))}$ algorithm, but a polynomial algorithm is still elusive. Notice we can derive Proposition 3.8 directly from Proposition 3.9 since all 4×4 matrices with only $\star$ on the main diagonal have polynomial list M-partition problems, with the exception of the matrix $2K_2$ and its complement.

3.2.2 Infinite classes of symmetric matrices

We survey here infinite classes of symmetric matrices M for which the complexity of M-partition has been classified. We restrict our attention to list M-partition problem since virtually all known results for "large" matrices rely on using lists. Here we focus on symmetric matrices, while the results for the more general matrices are presented in Section 3.3.2.

Much progress has been made in classifying the complexity of M when M is restricted

to be a matrix over a subset of $\{0, 1, \star\}$—i.e. one of $0, 1, \star$ is forbidden from M. Indeed when M is a matrix over $\{0, 1\}$, $\{0, \star\}$ or $\{1, \star\}$, the list M-partition problem is known to be **NP**-complete or polynomial.

We deal first with the class of all matrices over $\{0, 1\}$. In [28] the authors present the following simple classification.

Proposition 3.10 ([28]) *Let M be a symmetric matrix over $\{0, 1\}$ then the list M-partition problem is polynomial time solvable.*

The proof of this is a straightforward application of sparse-dense partitions. In fact we proved this result informally as an example near the end of Section 2.6. There is also a characterization of those graphs which admit an M-partition, for a $\{0, 1\}$ matrix M. A graph G is called a *minimal obstruction to M-partition* if G is not M-partitionable but for every $v \in V(G)$ the vertex deleted subgraph $G - v$ is M-partitionable. In [23] (cf. also [32]) it is shown that there are a finite number of minimal obstructions to M-partition, when M is a symmetric matrix over $\{0, 1\}$.

Proposition 3.11 ([23]) *Let M be a symmetric matrix over $\{0, 1\}$, with k zeros and l ones on the main diagonal. Then each minimal obstruction to M-partition has at most $(k + 1)(l + 1)$ vertices.*

In [32] there is a particularly simple proof which shows any minimal obstruction to M-partition has a bounded number of vertices.

Another broad class of matrices which has been studied is the class of all symmetric matrices over $\{0, \star\}$ (and by complement over $\{1, \star\}$). Recall that this class of matrices corresponds exactly to the H-colouring problem for symmetric digraphs H. In a series of papers [21, 24, 25] the complexity of the list H-colouring problem, for symmetric graphs H, has been completely classified as polynomial—in the case of so-called bi-arc graphs—and **NP**-complete otherwise. We will briefly describe the classification.

Let C be a circle, and let p and q be two diametrically opposed points on C. A *bi-arc representation* of a (symmetric) graph G is a family of ordered pairs of arcs, which associates for each $x \in V(G)$ an arc N_x which contains p but not q and an arc S_x which contains q but not p, and with the edges of G represented as follows:

if x and y are adjacent, then both $N_x \cap S_y = \emptyset$ and $N_y \cap S_x = \emptyset$,

if x and y are non-adjacent, then both $N_x \cap S_y \neq \emptyset$ and $N_y \cap S_x \neq \emptyset$.

(Note that in a bi-arc representation for any two pairs of intervals (N, S) and (N', S') we always have $N \cap S' = \emptyset \iff N' \cap S = \emptyset$.) A graph which admits a bi-arc representation is called a *bi-arc graph*.

Proposition 3.12 ([25]) *Let H be a symmetric digraph, then if H is a* bi-arc graph *the list H-colouring problem is polynomial; otherwise the list H-colouring problem is **NP**-complete.*

Corollary 3.13 *Let M be a symmetric matrix over $\{0, \star\}$, let H be the graph corresponding to M. Then the list M-partition problem is polynomial—when H is a bi-arc graph—and NP-complete otherwise.*

Note that, by complement the corollary also applies to matrices over $\{1, \star\}$. Surprisingly, bi-arc graphs are exactly those symmetric graphs which admit a conservative majority function[4, 24, 26]. Thus it can easily be decided in polynomial time if a fixed graph H is bi-arc—we simply enumerate all possible ternary functions of $V(H)^3 \to V(H)$ and check each to see if it the required conservative majority polymorphism.

There are more efficient ways of recognizing bi-arc graphs and also a characterization by forbidden subgraphs (described in [26]). We first describe *circular-arc graphs* and *associated bipartite graphs*. A graph H is a *circular-arc graph* if it is the intersection graph of a family of arcs on a circle. That is, for each $x \in V(H)$ we have an associated arc A_x, such that $xy \in E(H)$ if and only if $A_x \cap A_y \neq \emptyset$. For a graph H we define an *associated bipartite graph* to be the direct product $H \times K_2$, where K_2 is a clique on two vertices.

Proposition 3.14 ([26]) *Let H be a symmetric graph with associated bipartite graph $H \times K_2$. Then H is a bi-arc graph if and only if the complement of $H \times K_2$ is a circular-arc graph.*

Thus we can reduce deciding if H is a bi-arc graph to deciding if the complement of $H \times K_2$ is a circular-arc graph. Suppose G is a symmetric graph with n vertices and m edges. McConnell gives in [51] an $O(n+m)$ algorithm to decide if a graph G is a circular-arc graph, the algorithm also provides a circular-arc representation of G if one exists.

The complement of a bipartite graph is a *co-bipartite graph* and is said to be a *graph of clique-covering number two*. Using this terminology, Proposition 3.14 can be rephrased

as: H is bi-arc if and only if the complement of the associated bipartite graph $H \times K_2$ is a circular-arc graph of clique covering number two. Trotter and Moore [56], and more recently Feder, Hell and Huang [24], provide characterizations for circular-arc graphs of clique covering number two in terms of forbidden induced subgraphs.

If we restrict our attention to trees (with loops allowed), then there is a characterization of exactly which trees are bi-arc graphs directly in terms of a set of forbidden induced subgraphs for the graph H [25, 26].

Proposition 3.15 ([25, 26]) *Let H be a (symmetric) tree, with loops allowed. Then H is a bi-arc tree if and only if the subgraph induced by the loops of H is connected, and H does not contain, as an induced subgraph, any of the graphs shown in Figure 3.4.*

When M is allowed to contain entries from $\{0, 1, \star\}$, without restriction, much less is known. We are able to present a class of matrices M which have polynomial list M-partition problems. We can also exhibit others for which the list M-partition problems are **NP**-complete.

We begin with a class of matrices from [15] for which the list M-partition problem is solvable in polynomial time. We first recall the definition of skew partitions. A *skew partition* of a graph G is a partition into four (non)-empty parts A_1, A_2, B_1, B_2 so that A_1 and A_2 are completely adjacent and B_1 and B_2 are completely non-adjacent. This is in fact a matrix partition problem (Figure 3.5 shows the corresponding matrix).

One possible generalization of skew partitions is the following *extended skew partition problem*. An (n_1, n_2)-*extended skew partition* of a (symmetric) graph G is a partition of $V(G)$ into $(n_1 + n_2)$ non-empty parts $A_1, ..., A_{n_1}, B_1, ..., B_{n_2}$ such that there are all possible edges between the parts A_i and A_j (for distinct $i, j \in \{1, ..., n_1\}$) and no edges between the parts B_k and B_l (for distinct $k, l \in \{1, ..., n_2\}$). This is a matrix partition problem for matrix M with entries defined by $M_{ij} = 1$ if and only if $i \neq j$ and $i, j \in \{1, ..., n_1\}$, and $M_{ij} = 0$ if and only if $i \neq j$ and $i, j \in \{n_1 + 1, ..., n_1 + n_2\}$ ($M_{ij} = \star$ else). The matrix for $(3, 3)$-extended skew partition is shown in Figure 3.5.

The *extended skew partition problem* for a fixed matrix M, corresponding to an (n_1, n_2)-extended skew partition, is defined as follows. Decide if an input graph G admits an M-partition with all parts non-empty. This is a surjective M-partition problem. Dantas, et al. provide a polynomial algorithm for all (n_1, n_2)-extended skew partition problems in [15]—in fact their algorithm solves the more general list M-partition problem.

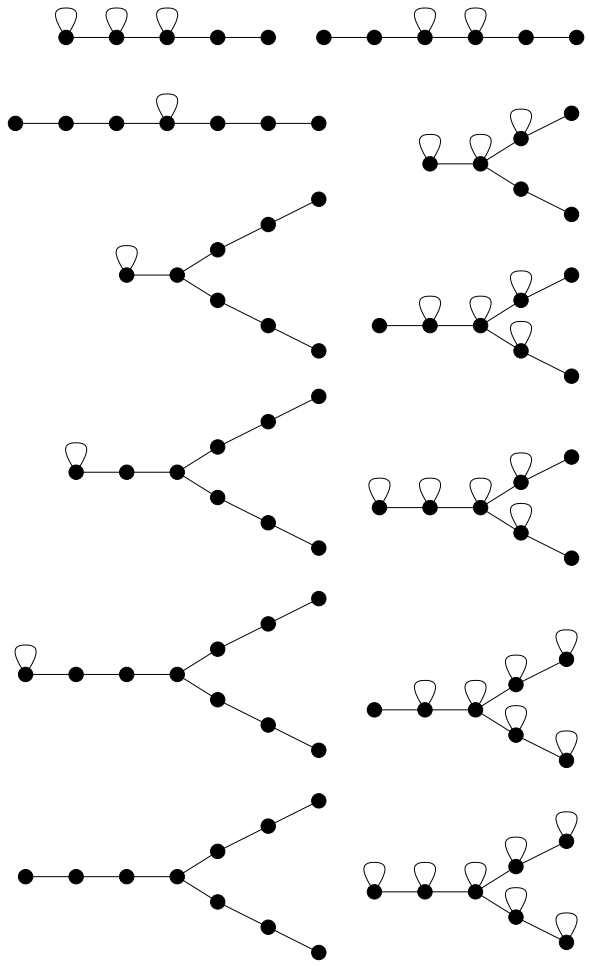


Figure 3.4: The twelve forbidden induced subgraphs for bi-arc trees.

Proposition 3.16 ([15]) *Let M be a symmetric matrix which corresponds to an (n_1, n_2)-extended skew partition problem. Then the list M-partition problem is polynomial time solvable.*

We now present a class of matrices for which the list M-partition problem is known to be **NP**-complete. Let H be a bipartite undirected graph which is *not* a bi-arc graph; by [24] we know that list H-colouring is **NP**-complete. We will construct a matrix M such that the list H-colouring problem can be reduced to the list M-partition problem.

Since H is bipartite we can assume that we have a partition of H into (H_A, H_B). Consider the matrix corresponding to H, this is a symmetric matrix M over $\{0, \star\}$. This matrix M can be partitioned into blocks. Let X be the diagonal submatrix corresponding to H_A, and

$$\begin{bmatrix} \star & 1 & \star & \star \\ 1 & \star & \star & \star \\ \star & \star & \star & 0 \\ \star & \star & 0 & \star \end{bmatrix} \qquad \begin{bmatrix} \star & 1 & 1 & \star & \star & \star \\ 1 & \star & 1 & \star & \star & \star \\ 1 & 1 & \star & \star & \star & \star \\ \star & \star & \star & \star & 0 & 0 \\ \star & \star & \star & 0 & \star & 0 \\ \star & \star & \star & 0 & 0 & \star \end{bmatrix}$$

Figure 3.5: The matrices for skew partition(left) and (3,3)-extended skew partition(right).

Y the diagonal submatrix corresponding to H_B. Let Z and its transpose Z^T be the off-diagonal submatrices which contain the remaining entries of M. Suppose H_A has k vertices and H_B has l vertices, then we will call Z the $k \times l$ *off-diagonal matrix corresponding to H.*

Proposition 3.17 ([29]) *Let H be a bipartite undirected graph which is* not *a bi-arc graph. Let Z be the $k \times l$ off diagonal matrix corresponding to H (or its complement). Let A be an arbitrary $k \times k$ matrix over $\{0, \star\}$ and let B be an arbitrary $l \times l$ matrix over $\{1, \star\}$. Let M be the $(l + k) \times (l + k)$ matrix composed of diagonal blocks A and B, and with off-diagonal blocks Z and Z^T. Then the list M-partition problem is **NP**-complete.*

The reduction is actually quite simple. Any graph G which admits a list homomorphism to H must also be bipartite, and can likewise be assumed to be pre-partitioned into (G_A, G_B). Given an input graph G with lists $L(G)$ obtain a new graph G' from G by joining each pair of vertices in G_B by an edge (the lists are not changed). Obviously G admits an H-colouring with respect to $L(G)$ if and only if G' admits an M-partition with respect to $L(G)$. This is an interesting class of matrices, since these list M-partition problems are **NP**-complete even when the input is restricted to *split graphs* (notice that the graph G' constructed above is a split graph).

3.3 Partitions of digraphs

We now summarize results for the more general case of M-partitions of digraphs. In this section we do not require the matrix M to be symmetric. We treat only the list M-partition problem since other variants of the M-partition problem have not yet been studied in this more general context.

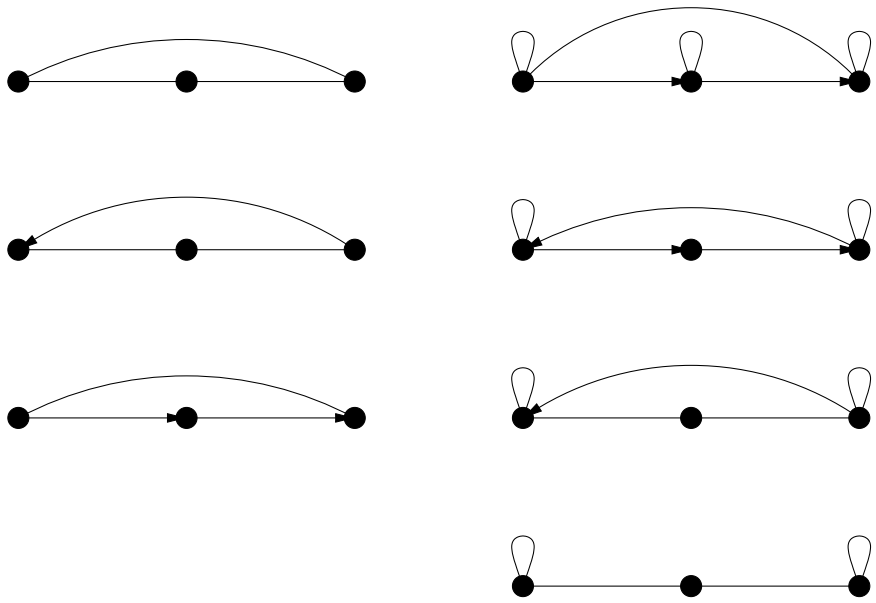


Figure 3.6: The seven **NP**-complete digraphs on at most 3 vertices.

3.3.1 Small matrices

We begin with the progress on small matrices. Feder, Hell and Tucker-Nally [31] (cf. also [57]) classify the complexity of all list H-colouring problems, where H is a digraph on at most 3 vertices. This yields a classification of the complexity of all matrices over $\{0, \star\}$ (and by complement $\{1, \star\}$) with size $m \leq 3$.

Proposition 3.18 ([31]) *Let H be a digraph with at most 3 vertices, then if H is one of the digraphs shown in Figure 3.6 then the list H-colouring problem is **NP**-complete; otherwise the list H-colouring problem is polynomial.*

The authors, then show that all remaining matrices of size $m \leq 3$ are solvable in polynomial time. Analogously to Proposition 3.6 we have the following dichotomy result which classifies completely the complexity of the list M-partition problem as **NP**-complete or polynomial.

Proposition 3.19 ([31]) *Let M be a (not necessarily symmetric) matrix over $\{0, 1, \star\}$, with size $m \leq 3$. Then if M is a matrix corresponding to a digraph in Figure 3.6(or the complement of such a matrix) then the list M-partition problem is **NP**-complete; otherwise the list M-partition problem admits a polynomial algorithm.*

This is the extent of current knowledge for small (not necessarily symmetric) matrices.

3.3.2 Infinite classes of matrices

We now survey the progress on infinite families of matrices. We first introduce what is known for matrices over $\{0,1\}$,$\{0,\star\}$ and $\{1,\star\}$. We then present what is known for the more general case when M is allowed to contain all elements from $\{0,1,\star\}$.

We discussed near the end of Section 2.6 the case where M is an arbitrary matrix over $\{0,1\}$. That argument is a very slight modification of a result in [28] which dealt with symmetric matrices.

Proposition 3.20 ([28]) *Let M be a matrix over $\{0,1\}$ then the list M-partition problem is polynomial time solvable.*

As in the case of symmetric matrices, when M is a matrix over $\{0,1\}$ there is a finite set of minimal M-obstructions. A bound on the size can be obtained by a slight modification to the proof of [32] for the symmetric case.

Proposition 3.21 *Let M be an $m \times m$ matrix over $\{0,1\}$. Then there is a constant c such that any minimal M-obstruction has at most c vertices.*

For matrices over $\{0,\star\}$—similar to Proposition 3.12—there is a dichotomy for list M-partition. If M is a matrix over $\{0,\star\}$ then the list M-partition problem is polynomial or **NP**-complete. This is a special case of the more general result of Bulatov [7, 8, 9] about *conservative constraint satisfaction problems* discussed in Section 2.1. Recall that Proposition 2.1 shows that all conservative constraint satisfaction problems are **NP**-complete or polynomial. As a special case we get a dichotomy for list homomorphism to digraphs which in turn implies a dichotomy for list M-partition problems when M is a matrix over $\{0,\star\}$ or $\{1,\star\}$.

Proposition 3.22 ([7]) *Let H be a digraph (with loops allowed), then the list H-colouring problem is polynomial if for each two-element subset $\{x,y\} \subseteq V(H)$ there exists a conservative polymorphism f such that the restriction $f|_{\{x,y\}}$ is a* semi-lattice*, a* majority*, or a* Mal'tsev *operator on the subset $\{x,y\}$; otherwise the list H-colouring problem is **NP**-complete.*

Corollary 3.23 *Let M be a (not necessarily symmetric) matrix over $\{0,\star\}$ or $\{1,\star\}$ then the list M-partition problem is polynomial or **NP**-complete.*

Proposition 3.22 can be used to test if the list H-colouring problem is polynomial or **NP**-complete. We can enumerate all binary and ternary operators over $V(H)$, then for each two-element subset search for a suitable polymorphism. Since H is fixed there are only constantly many binary and ternary polymorphism to consider. However, unlike the case for symmetric graphs—Proposition 3.12—there is no graph theoretic characterization to go along with the dichotomy. A simple characterization would be quite desirable, since the test for polynomiality suggested above involves *huge* constants.

We now turn our attention to the case where M is not restricted to contain only proper subsets of $\{0,1,\star\}$. As was the case for symmetric matrices, we know much less about the list M-partition problem in this more general setting.

In [30] Feder, Hell, Král and Sgall presented a class of matrices which admit polynomial algorithms for list M-partition. This is the class of matrices with the so-called *bisplit property*. This appears to be the only large class of general matrices whose complexity has been classified.

Before defining the bisplit property we will first describe several classes of matrices which share this property. Recall that a *split graph* is a graph which admits a partition into a clique and an independent set. Split graphs are recognizable in linear time [37]. In [3] Brandstädt et al. generalized this concept to so-called *bi-split graphs*. Bisplit graphs are those (symmetric) graphs G which admit a partition into three non-empty independent sets V_1, V_2 and V_3 such that V_2 and V_3 induce a complete bipartite graph in G. Brandstädt, et al. also introduce *k-bisplit graphs*, which are (symmetric) graphs G which admit a partition into $2k+1$ non-empty independent sets $V_1, ..., V_{2k+1}$ such that V_{2i} and V_{2i+1} induce a complete bipartite graph in G, for $i = 1, .., k$. These are clearly matrix partition problems (with symmetric matrices). Some example matrices are shown in Figure 3.7.

Feder, et al. have further generalized this notion to *(A,B)-bisplit matrices* and provided a polynomial algorithm for any (A,B)-bisplit partition problem in [30]. A matrix M is an (A,B)-bisplit matrix if it can be constructed as follows. Let A and B be two (not necessarily symmetric) matrices over $\{0,1\}$ of size k and l respectively. Then construct M to be the $(k+l) \times (k+l)$ matrix which contains A and B as (independent) diagonal submatrices, and all other entries $\star$'s. Clearly, the matrices corresponding to recognition of *split graphs*,

$$\begin{bmatrix} 0 & \star \\ \star & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & \star & \star \\ \star & 0 & 1 \\ \star & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & \star & \star & \star & \star \\ \star & 0 & 1 & 0 & 0 \\ \star & 1 & 0 & 0 & 0 \\ \star & 0 & 0 & 0 & 1 \\ \star & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 3.7: The matrices for split partition(left), bisplit partition(centre) and 2-bisplit partition(right).

bisplit graphs and *k-bisplit graphs* are (A, B)-bisplit matrices.

All (A, B)-bisplit matrices share a common property, called the *bisplit property* in [30]. Let M be an $m \times m$ matrix. For a fixed $i \in \{1, ..., m\}$ we define the i^{th} *cross* of M to be the pair of ordered sets (R_i, C_i) where $R_i = \{M_{ic} \mid c = 1, ..., m\}$ is the i^{th} row of M and $C_i = \{M_{ri} \mid r = 1, ..., m\}$ is the i^{th} column. Two parts i and j are said to be *independent* if the set of $\star$ entries in the corresponding crosses are incomparable by inclusion. That is, one of the following is true:

- there exist s and t such that for the ordered sets R_i and R_j we have $M_{is} = \star \neq M_{js}$ and $M_{it} \neq \star = M_{jt}$, or
- there exist s and t such that for the ordered sets C_i and C_j we have $M_{si} = \star \neq M_{sj}$ and $M_{ti} \neq \star = M_{tj}$.

A matrix M has the *bisplit property* if M contains no set of three pairwise independent parts.

Proposition 3.24 ([30]) *Let M be a (not necessarily symmetric) matrix over $\{0, 1, \star\}$ with the* bisplit property*, then the list M-partition problem is polynomially solvable.*

Clearly (A, B)-bisplit matrices have the bisplit property since at most two parts can be mutually independent (all $\star$ entries form symmetric rectangles). The examples of (A, B)-bisplit shown in Figure 3.7 are symmetric matrices, however we remark that a matrix M need not be symmetric to have the bisplit property.

Other than matrices with the bi-split property (and the symmetric classes presented earlier) we do not know of any other large class of matrices with entries from $\{0, 1\star\}$ whose complexity has been classified. In the remainder of this thesis we classify the complexity of the list M-partition problem, for a new infinite class of matrices. We show that there is

a dichotomy for the list M-partition problem when M is a tridiagonal matrix. Each such problem can be classified as either polynomial or **NP**-complete. Our results rely heavily on results for conservative constraint satisfaction problems, and thus it is convenient to phrase things in terms of relations. Thus, for the remainder of this thesis, we use the trigraph homomorphisms model of M-partitions.

Chapter 4

Trigraph Paths

In this chapter we prove a dichotomy for the complexity of the list H-colouring problem for *trigraph paths* H (Theorem 4.2). Each such problem can be classified as either polynomial or **NP**-complete. Recall that a *trigraph path* is a trigraph H such that the underlying undirected graph induced by $E(H)$ is a disjoint union of paths (with loops allowed). The adjacency matrix of a trigraph path is a tridiagonal matrix; thus the trigraph paths discussed in this chapter correspond directly to tridiagonal matrices.

A note on conventions. Recall that for a binary relation R over a set X we say that x *is adjacent with* y *in* R if at least one of xy or yx is in R—otherwise we say that x *is non-adjacent with* y *in* R. Recall that trigraphs have two adjacency relations $E(H)$ and $N(H)$. For the sake of brevity, when discussing trigraphs H, we will use the term *adjacent* in place of *adjacent in* $E(H)$, and *non-adjacent* in place of *non-adjacent in* $E(H)$. This should not cause much confusion, since it matches the convention we adopted earlier for showing diagrams of trigraphs. In general, we consider the list H-colouring problem with representatives and denote the representative of a vertex x by v_x.

We define for a trigraph H an *associated digraph* H^- as follows. Let $V(H^-) = V(H)$ and let $E(H^-) = \{xy \in E(H) \mid xx, yy, xy, yx \in N(H)\}$. That is, $E(H^-)$ contains exactly those edges $xy \in E(H)$ such that neither x nor y is a strong loop, and neither xy nor yx is a strong edge. We may think of H^- as being obtained from H by deleting all strong loops along with the incident edges, and deleting all edges xy such that at least one of xy or xy is a strong edge. Note that H^- is a digraph with loops allowed. Figure 4.1 shows an example of a trigraph H and its associated digraph H^-.

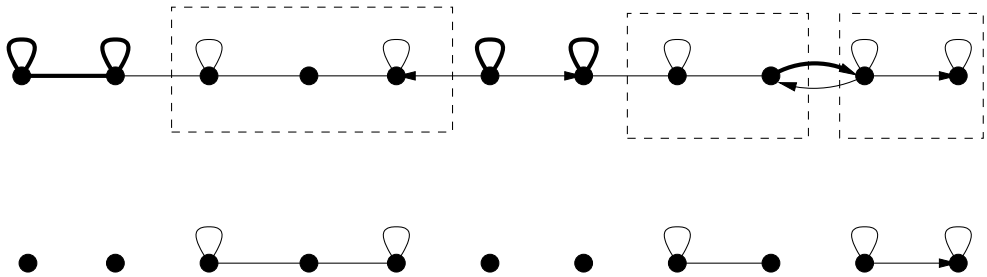


Figure 4.1: A trigraph path H is shown above. The associated digraph H^- is shown below.

Theorem 4.1 *Let H be a trigraph with associated digraph H^-. If the list H^--colouring problem for the associated digraph H^- is **NP**-complete then the list H-colouring problem is also **NP**-complete.*

Proof. We will assume the list H^--colouring problem is **NP**-complete. We may assume, without loss of generality, that there is some connected component H_1^- of H^- for which list colouring is **NP**-complete. Consider H_1 the sub-trigraph of H induced on the vertices $V(H_1^-)$. The trigraph H_1 contains no strong edges implying that $N(H_1) = V(H_1) \times V(H_1)$ and $E(H_1) = E(H_1^-)$. Thus the list H_1- and H_1^--colouring problems are equivalent (see the discussion of Proposition 3.2). This implies that H-colouring is **NP**-complete even when the input is restricted so that all lists contain only vertices of $V(H_1^-)$; and so, if list H^--colouring is **NP**-complete then list H-colouring is also **NP**-complete. □

As an example notice that the list H-colouring problem for the trigraph depicted in Figure 4.1 is **NP**-complete, since the associated digraph H^- contains, as an induced subgraph, the digraph corresponding to the stable-cutset problem (the connected component with three vertices).

In the next few sections we show that, for trigraph paths H, a polynomial list H^--colouring problem implies a polynomial list H-colouring problem, yielding the following theorem.

Theorem 4.2 *Let H be a trigraph path with associated digraph H^-. The list H-colouring problem is polynomial if the list H^--colouring problem is polynomial, and is **NP**-complete otherwise.*

We begin, in the next section, with some useful observations about representatives and arc-consistency. In particular, Lemma 4.3, will be of considerable use later.

4.1 Consequences of arc-consistency with representatives

We consider an instance $G, L(G)$ of the list H-colouring problem with representatives. We can often use representatives to obtain reduced lists with nice properties. Recall that a set of lists $L^*(G)$ are called reduced lists for $L(G)$ if both the following conditions are met. First, for each $v \in V(G)$ we have $L^*(v) \subseteq L(v)$. Secondly, there is a homomorphism of $G \to H$ with respect to the lists $L^*(G)$ if and only if there is a list homomorphism of $G \to H$ with respect to the original lists $L(G)$. We remark that the following lemmas can also be derived from the arc-consistency procedure for trigraphs described in Section 2.4.

Lemma 4.3 *Let x and y be distinct vertices of H such that at least one of xy or yx is a strong edge of H. Let v be adjacent with the representative of x. Then the list $L(v)$ can be reduced to a list $L^*(v)$, such that $|L^*(v)| \leq 2$.*

Proof. Let $f : G \to H$ be a homomorphism with respect to $L(G)$. Without loss of generality assume $y = x + 1$, and let v_x and v_{x+1} be the representatives of x and $x + 1$ respectively.

There are two cases. Consider first that v is adjacent with both of v_x and v_{x+1}. Then f must map v to a common neighbour of both x and $x + 1$. Since H is a trigraph path, the common neighbours of x and $x + 1$ are contained in the set $\{x, x + 1\}$. Thus we can set $L^*(v) = L(v) \cap \{x, x + 1\}$.

Now, without loss of generality, consider the case that v is adjacent with v_x but is non-adjacent with v_{x+1}. Then f must map v to a neighbour of x. Since H is a trigraph path, the neighbours of x are contained in the set $\{x - 1, x, x + 1\}$. But f cannot map v to x, since then a non-edge will be mapped across the strong edge $(x, x + 1)$. Therefore we can set $L^*(v) = L(v) \cap \{x - 1, x + 1\}$.

Since we obtained $L^*(v)$ by removing elements from $L(v)$ which are inconsistent with any homomorphism, we have $G \to H$ with respect to $L^*(v)$ if and only if $G \to H$ with respect to $L(v)$.□

Let x be a strong loop of H—i.e. $xx \in E(H) \setminus N(H)$. If x is not adjacent with any strong loop, we say that x is an *isolated strong loop*. Otherwise we say x is a *non-isolated strong loop*—i.e. at least one of $x - 1$, $x + 1$ is also a strong loop of H.

Lemma 4.4 *Let x be a non-isolated strong loop of H with representative $v_x \in V(G)$. Let v be adjacent with v_x. Then the list $L(v)$ can be reduced to a list $L^*(v)$, such that $|L^*(v)| \leq 2$.*

Proof. Let $f : G \to H$ be a homomorphism with respect to $L(G)$. Let x be a non-isolated strong loop, and without loss of generality let $x + 1$ be an adjacent strong loop.

Let $v \in V(G)$ be adjacent with v_x, then f maps v to a neighbour of x—since H is a trigraph path, $f(v) \in \{x - 1, x, x + 1\}$. There are two cases to consider. First suppose v is non-adjacent with v_{x+1}, then f cannot map v to $x+1$ and we can set $L^*(v) = L(v) \cap \{x-1, x\}$. Now, suppose that v is adjacent with v_{x+1}, then f maps v to a neighbour of $x + 1$. This excludes $f(v) = x - 1$, and so we can set $L^*(v) = L(v) \cap \{x, x + 1\}$.□

Lemma 4.5 *Let x be an* isolated *strong loop of H, and let $x - 1$ and $x + 1$ be irreflexive. For each graph G on n vertices there exists a family of $p \leq n$ cliques $\mathcal{C} = \{C_1, ..., C_p\}$ such that there is a homomorphism of $G \to H$ if and only if there is a homomorphism f such that either $f^{-1}(x) = C_i$, for some i, or $f^{-1}(x) = \emptyset$.*

Proof. The proof is by the sparse-dense method. Let x be an isolated strong loop, and let $x - 1$ and $x + 1$ be loopless. Let v_x be the representative of x. (If x does not have a representative then x appears on no list in $L(G)$ and we are done.)

Let $f : G \to H$ be a homomorphism with respect to $L(G)$. Let G_x be the subgraph of G induced by the closed neighbourhood of v_x. The pre-image $f^{-1}(x)$ contains only vertices from $V(G_x)$. The homomorphism f partitions G_x into two parts, a clique $f^{-1}(x)$, and an independent set $f^{-1}(x - 1) \cup f^{-1}(x + 1)$; that is, G_x must be a split graph.

Split graphs have some nice properties. First, split graphs are efficiently recognizable in polynomial time by examining the degree sequence of a graph [37] (recognition remains efficient even with lists, cf. [38, 28]). Second, there are at most a linear number of partitions of a split graph into a clique and independent set. This is because cliques and independent sets are sparse-dense with respect to each other; since a graph in both classes has at most one vertex. Thus we can apply Proposition 2.8 for the classes of cliques and independent sets with constant $c = 1$ and get that there are at most n sparse-dense partitions of a graph on n vertices.

So, first we check that G_x is a split graph, then we enumerate in polynomial time all possible partitions of G_x into a clique and an independent set. The pre-image of x under any homomorphisms must be one of these cliques. □

Suppose H has some vertices $x_1, x_2, ..., x_k$ which are isolated strong loops. The lemma tells us that for each of these isolated strong loops there is a number of cliques bounded by

some polynomial in the size of G—suppose $f(n)$. The number of different ways to assign cliques to $x_1, x_2, ..., x_k$ is bounded by $f(n)^k$, a polynomial in n. Thus we can try all possible assignments of cliques to isolated strong loops in polynomial time.

In each such assignment we have associated with each isolated strong loop $x \in V(H)$ a clique of G which maps to x. Let C_x the be *clique associated with* x. Notice that by assigning C_x to x we prevent any other vertex of G from mapping to x—thus we can remove x from all lists $L(v)$ for $v \notin V(C_x)$ and set the lists $L^*(v) = \{x\}$ for $v \in V(C_x)$.

Lemma 4.6 *Let x be an isolated strong loop of H, and let $x-1$ and $x+1$ be irreflexive. Let C_x be the clique associated with x. Let $v \in V(G)$ be a vertex which is adjacent with a vertex in C_x. Then the list $L(v)$ can be reduced to a list $L^*(v)$, such that $|L^*(v)| \leq 2$.*

Proof. Let f be a homomorphism of $G \to H$ with respect to $L(G)$ such that $f^{-1}(x) = C_x$. Let $v \notin V(C_x)$ be a vertex which neighbours $u \in V(C_x)$. Then $f(v)$ must be a neighbour of x in H. Thus we can set $L^*(v) = L(v) \cap \{x-1, x+1\}$. □

4.2 Trigraph paths without strong loops

In this section we prove Theorem 4.2 in the special case that H is a trigraph path without strong loops; in subsequent sections we show how to remove this restriction. Thus we consider trigraph paths H without strong loops and prove the following theorem.

Theorem 4.7 *Let H be a trigraph path without strong loops with associated digraph H^-. Then the list H-colouring problem is polynomial if the list H^--colouring problem is polynomial and is **NP**-complete otherwise.*

Recall from Proposition 3.22 that each list H^--colouring problem is polynomial or **NP**-complete. In light of Theorem 4.1, we can assume list H^--colouring is polynomial. As before we consider a fixed instance $G, L(G)$ of the list H-colouring problem with representatives (cf. Lemma 2.4). We denote the representative of a vertex x by v_x. We assume that the vertices of H are ordered so that adjacent vertices are consecutive. We highlight again that we say two vertices x and y are *adjacent* in H if at least one of xy or yx is an edge in $E(H)$.

We define the *neighbourhood of a strong edge xy* to be the set $P(xy) = \{p_1, p_2, p_3, p_4\}$ where p_1, p_2, p_3, p_4 are four consecutive vertices of H such that $xy = p_2p_3$ is the strong edge.

Note that, since we consider a path with no strong loops, we have $x \neq y$. If x or y is an endpoint of the path we can omit the relevant vertex from $P(xy)$—e.g. if the only neighbour of x is y then there is no p_1 in $P(xy)$. For ease of exposition when considering $P(xy)$ we will without loss of generality ignore the case that x or y is an endpoint of the path.

It will be convenient to mark a subset of $V(G)$ as special. A vertex $v \in V(G)$ is a *blue vertex* of G if v is adjacent with one of the representatives,v_x or v_y, of some strong edge xy. We say in this case that v is a *blue vertex with type* xy. Note that it is possible a vertex may have more than one type. If two *blue* vertices u and v have at least one type in common we say that *u and v are blue vertices with the same type.* The blue vertices with type xy are exactly those vertices of G which can map to the strong edge xy in a list homomorphism.

Lemma 4.8 *Let $f : G \to H$ be a list homomorphism with respect to $L(G)$. If $f(u)f(v)$ is a strong edge of H, then u and v are both* blue vertices with the same type.

Proof. Let xy be a strong edge and suppose that $f(u) = x$ and $f(v) = y$. Let $\Gamma(v_x)$ and $\Gamma(v_y)$ be the closed neighbourhoods of the representatives v_x and v_y respectively. Since xy is a strong edge we have that $f^{-1}(x) \subseteq \Gamma(v_y)$ and $f^{-1}(y) \subseteq \Gamma(v_x)$. This implies that u and v are both *blue vertices with type xy.*□

We will now construct edge-coloured digraphs H^* and G^*, on $V(H)$ and $V(G)$ respectively. We also construct a set of lists $L^*(G)$ such that there is a list H-colouring of G with respect to $L(G)$ if and only if there is a list homomorphism of G^* to H^* with respect to $L^*(G)$.

Before constructing our H^* and G^* we provide some intuition for the construction. We know from Lemma 4.8 that only pairs of blue vertices with the same type can map to strong edges of H. Suppose u and v are *blue* vertices with the same type, then all possible mappings of the ordered pair uv into H are given by $L(u) \times L(v)$. If we suppose $uv \in E(G)$, then $(L(u) \times L(v)) \cap E(H)$ gives the mappings in H which are consistent with a homomorphism. Likewise, if $uv \notin E(G)$, then $(L(u) \times L(v)) \cap N(H)$ gives mappings which are consistent with a homomorphism. Recall that we made this observation previously, in Section 2.5 for the case of lists of size at most two. The idea is to treat edges (non-edges) between pairs of blue vertices with the same type independently of the rest of G. We want to remove the strong edge xy, and replace it with new relations which put constraints only on blue vertices with type xy.

Let v be a blue vertex with type xy, then by Lemma 4.3 we can obtain a reduced list $L^*(v)$ of size at most two. It is easy to see from the proof of Lemma 4.3 that $L^*(v)$ must be a subset one of the following three lists from the neighbourhood $P(xy)$: $\{p_1, p_3\}, \{p_2, p_3\}$, or $\{p_2, p_4\}$.

We now construct the edge coloured digraph H^*. Let $V(H^*) = V(H)$ and define edge relations for H^* as follows. For each strong edge xy, and each pair of (not necessarily distinct) sets L, L' from $\{p_1, p_3\}, \{p_2, p_3\}, \{p_2, p_4\}$, construct the two relations

$$E^{L,L'}(H^*) = (L \times L') \cap E(H), \text{ and}$$

$$N^{L,L'}(H^*) = (L \times L') \cap N(H).$$

We also construct a relation $W(H^*)$

$$W(H^*) = \{xy \in E(H) \mid xx, yy, xy, yx \in N(H)\}$$

The relation $W(H^*)$ is exactly the relation $E(H^-)$ introduced earlier. Figure 4.1 shows an example of $W(H^*)$ for a trigraph path H.

We also construct an edge-coloured digraph G^*. Let $L^*(G)$ be a set of reduced lists such that each *blue* vertex v has a list with $|L^*(v)| \leq 2$. Let $V(G^*) = V(G)$ and define edge relations for G^* as follows. For each strong edge xy, and each pair of (not necessarily distinct) sets L, L' from $\{p_1, p_3\}, \{p_2, p_3\}, \{p_2, p_4\}$, construct the two relations

$$E^{L,L'}(G^*) = \{uv \in E(G) \mid u, v \text{ are blue with type } xy, \text{ and } L^*(u) \subseteq L, L^*(v) \subseteq L'\}, \text{ and}$$

$$N^{L,L'}(G^*) = \{uv \notin E(G) \mid u, v \text{ are blue with type } xy, \text{ and } L^*(u) \subseteq L, L^*(v) \subseteq L',\ u \neq v\}.$$

Finally, we construct a relation $W(G^*)$ as follows.

$$W(G^*) = \{uv \in E(G) \mid u, v \text{ are not both } \textit{blue} \text{ vertices with the same type}\}$$

The relation $W(G^*)$ contains the edges of G which *cannot* map to a strong edge of H.

Lemma 4.9 *Let $f : V(G) \to V(H)$ be a mapping of G to H. Then f is a homomorphism of G to H with respect to $L(G)$ if and only if f is a homomorphism of G^* to H^* with respect to $L^*(G)$.*

Proof. Since the lists $L^*(G)$ are reduced lists for $L(G)$, we have that $G \to H$ with respect to $L(G)$ if and only if $G \to H$ with respect to $L^*(G)$. We will show $G \to H$ with respect to $L^*(G)$ if and only if $G^* \to H^*$ with respect to $L^*(G)$.

Let $f : G^* \to H^*$ be a homomorphism of G^* to H^* with respect to the lists $L^*(G)$. Let uv be a pair of distinct vertices from G. Consider first that both of u and v are *blue* vertices with the same type in G—suppose they have type xy. Then by Lemma 4.3 both u and v have lists of size at most two from the neighbourhood $P(xy)$. Thus $E^{L(u),L(v)}$ and $N^{L(u),L(v)}$ are relations in H^*, and so $f(u)f(v)$ is in one of $E^{L(u),L(v)} \subseteq E(H)$ or $N^{L(u),L(v)} \subseteq N(H)$. In any case an edge (non-edge) between u and v is preserved by f. Now, suppose u and v are not both *blue* vertices with the same type. If $uv \in E(G)$, then by construction we know that $uv \in W(G^*)$ and we have $f(u)f(v) \in W(H^*) \subseteq E(H)$. The last case to consider is the case that $uv \notin E(G)$. In this case, Lemma 4.8 implies that $f(u)f(v)$ is a not strong edge and so $f(u)f(v) \in N(H)$ as required.

Now, let $g : G \to H$ be a homomorphism of G to H with respect to the lists $L^*(G)$. We verify that g is also a homomorphism of G^* to H^* with respect to $L^*(G)$. Consider first $uv \in E^{L,L'}(G^*)$. By construction we have $uv \in E^{L,L'}(G^*)$ only if $uv \in E(G)$ and $L(u) \subseteq L$ and $L(v) \subseteq L'$. All possible mappings of the pair uv into $E(H)$ are given by $L(u) \times L(v) \cap E(H)$, but this is a subset of $E^{L,L'}(H^*)$. Thus $g(u)g(v) \in E^{L,L'}(H^*)$ as required. A similar argument shows that for $uv \in N^{L,L'}(G^*)$ we have $g(u)g(v) \in N^{L,L'}(H^*)$. It remains to verify that W is preserved under g, so consider a pair $uv \in W(G^*)$. By Lemma 4.8 neither $f(u)f(v)$ nor $f(v)f(u)$ is a strong edge of H, thus $f(u)f(v)$ is in $W(H^*)$ as required. □

So, we have reduced the list H-colouring problem to the list H^*-colouring problem. In the remainder of this section we show that the complexity of the list H^*-colouring problem depends entirely on the the relation $W(H^*)$, that is the digraph H^-. We have shown in Theorem 4.1 that if list H^--colouring is **NP**-complete then list H^*-colouring is also **NP**-complete. What remains to show is that a polynomial list H^--colouring problem implies a polynomial list H^*-colouring problem.

We begin with a case which is easily described. Suppose first that H^- has a conservative majority polymorphism, it then follows easily that H^* has a conservative majority polymorphism. This is since each relation $E^{L,L'}(H^*)$ and $N^{L,L'}(H^*)$ is a subset of a product of two sets of size at most two; thus, by Lemma 2.3, any ternary conservative function which is a majority operator is a majority polymorphism of these additional relations. So, in this instance, the majority polymorphism of H^- is also a majority polymorphism of H^*. Therefore, the list H^*-colouring problem is polynomial.

In general things can be more complicated, but the essence is similar. Recall from Proposition 2.1 that list H^--colouring is polynomial if there is a collection Φ of conservative polymorphisms of H^- such that for each two-element subset $\{x, y\} \subseteq V(H)$ there exists a polymorphism $\varphi \in \Phi$ such that the restriction $\varphi|_{\{x,y\}}$ is a semi-lattice, majority or Mal'tsev operator. We will first define such a collection Φ of polymorphisms for H^-, then show that each $\varphi \in \Phi$ is also a polymorphism of H^*. This shows that if list H^--colouring is polynomial then list H^*-colouring is also polynomial.

Since we have no strong loops, then H^- is obtained from H by deleting all edges xy such that at least one of xy or yx is a strong edge. Let $H_1^-, H_2^-, ..., H_l^-$ be the connected components of H^-. Since list H^--colouring is polynomial, then by Proposition 2.2 for each connected component H_i^- there exist conservative polymorphisms f_i, g_i, h_i such that for each two-element subset $\{a, b\}$ of $V(H_i^-)$ one of the following is true:

$f_i|_{\{a,b\}}$ is a semi-lattice operator,

$g_i|_{\{a,b\}}$ is a majority operator, or

$h_i|_{\{a,b\}}$ is a Mal'tsev operator.

We can extend each of f_i, g_i and h_i to be conservative polymorphisms on the entire digraph H^- in an obvious way by setting $f_i(x_1, x_2) = x_1$ if at least one of x_1, x_2 is not in $V(H_i^-)$, likewise $g_i(x_1, x_2, x_3) = h_i(x_1, x_2, x_3) = x_1$ if at least one of x_1, x_2, x_3 is not in $V(H_i^-)$.

It is straightforward to show that f_i, g_i and h_i are polymorphisms; we will discuss the case of f_i, the other cases follow similarly. We verify that the extension of f_i described above takes edges of $H^- \times H^-$ to edges of H^-. It is sufficient to consider each connected component of $H^- \times H^-$ separately. We claim that each connected component of $H^- \times H^-$ is a subset of some smaller product $H_j^- \times H_k^-$ where H_j^- and H_k^- are (not necessarily distinct) connected components of H^-. This follows from the definition of direct products, since we have an edge $(x_1y_1, x_2y_2) \in E(H^- \times H^-)$ only if x_1x_2 is an edge in some connected component H_j^- and y_1y_2 is an edge is some connected component H_k^-. Thus, consider $f_i : V(H_j^-) \times V(H_k^-) \to V(H^-)$. If we have $j = k = i$ then, by definition, f_i is a homomorphism. In all other cases f_i is a projection onto $V(H_j^-)$, and is thus also a homomorphism. Therefore the extended f_i is a polymorphism of H^-.

Each of the polymorphisms f_i, g_i and h_i for $i = 1, .., l$ is a projection on the two-element sets $\{a, b\} \subseteq V(H)$ where a and b are in different connected components of H^-. In order to

satisfy the conditions of Proposition 2.1 we need a polymorphism which is a majority, semi-lattice, or Mal'tsev operator on these two-element sets. Thus we will define a conservative polymorphism π which is a majority operator on all such two-element sets.

For $x \in H_i^-$ and $y, z \in H_j^-$, where $i \neq j$, we set

$$\pi(x, y, z) = \pi(y, x, z) = \pi(y, z, x) = y, \text{ and}$$

$$\pi(x, z, y) = \pi(z, x, y) = \pi(z, y, x) = z.$$

That is, we ignore x and take π to be the first (with respect to order of arguments) of y or z to appear—i.e. π is essentially a binary operator on the set $\{y, z\}$. Our definition guarantees that π is a majority operator on every two-element set $\{a, b\}$ with a and b in distinct connected components. Otherwise, if x, y, z are in three distinct components of H^- or exactly one connected component, we set π to be the projection onto its first co-ordinate—for example $\pi(x, y, z) = x$ and $\pi(y, z, x) = y$. It is probably worth emphasizing that π is *not* a majority operator on $V(H)$, it is only a majority operator on the two-element subsets $\{a, b\} \subseteq V(H)$ such that a and b are in different connected components of H^-.

It is straightforward to check that π is a polymorphism of H^-. Indeed, suppose we have $x_1 \in H_1^-$, $x_2 \in H_2^-$ and $x_3 \in H_3^-$; then in the product $(H^-)^3$ the vertex (x_1, x_2, x_3) is adjacent only with vertices (x_1', x_2', x_3') such that $x_1' \in H_1^-$, $x_2' \in H_2^-$ and $x_3' \in H_3^-$. Thus in checking that π takes edges of $(H^-)^3$ to edges of H^- it suffices to consider the two cases of the definition of π separately. First, consider x_1, x_2, x_3 in three distinct connected components or all in the same component, then $\pi(x_1, x_2, x_3)$ is a projection, and thus obviously a polymorphism. Second consider the case $x \in H_i^-$ and $y, z \in H_j^-$, where $i \neq j$, then π is essentially a binary operator which takes y, z and returns the projection onto the first coordinate and so is also a polymorphism.

The set $\Phi = \pi \cup \{f_i, g_i, h_i \mid i = 1, ..., l\}$ is a collection of conservative operators such that for each two-element subset $B \subseteq V(H)$ there exists $\varphi \in \Phi$ such that the restriction $\varphi|_B$ is a semi-lattice, majority, or Mal'tsev operator. We have seen that each $\varphi \in \Phi$ is a polymorphism of $E(H^-)$. The final step in proving Theorem 4.7 is to show that each φ is also a polymorphism of the edge-coloured digraph H^*.

Since $W(H^*) = E(H^-)$ it is immediate that all $\varphi \in \Phi$ are polymorphisms of $W(H^*)$. We thus need only verify that each $\varphi \in \Phi$ is a polymorphism of the additional relations $E^{L,L'}(H^*)$ and $N^{L,L'}(H^*)$, for each pair of sets L, L' as defined in the construction. It will

be sufficient to consider only the relations added for some fixed strong edge xy, we can apply the same arguments for any strong edge of H.

Lemma 4.10 *Let p_1, p_2, p_3, p_4 be four consecutive vertices of H^-, such that $xy = p_2p_3$ is a strong edge of H. Let Φ be the collection of polymorphism defined above. Let L, L' be sets from $\{p_1, p_2\}, \{p_2, p_3\}, \{p_2, p_4\}$ and let $R \subseteq L \times L'$ be a relation. Then $\varphi \in \Phi$ is a polymorphism of R.*

Proof. Let D be the digraph with vertex set $V(H)$ and edge relation $R(D) = R$. Note that D is a digraph with loops allowed. The relation R is a subset of a product of two sets of size at most two. Recall from the proof of Lemma 2.3 that to check that an operator φ is a polymorphism of such a relation we only need to consider the value of φ on subsets of size at most two. In fact, it suffices to consider the values of φ on subsets $\{a, b\}$ such that either $a, b \in L$ or $a, b \in L'$, since all edges of R are directed from L to L'.

Suppose first $L = L'$, then there are at most two non-isolated vertices in R, and further they are in distinct connected components of H^-. It is easy to see that φ is a polymorphism, since by definition it is either a projection (if $\varphi \neq \pi$) or a majority operator (if $\varphi = \pi$).

Thus, suppose $L \neq L'$. Without loss of generality, suppose $L = \{p_1, p_3\}$. This implies that p_1 is in $L \setminus L'$, and p_2 is in $L' \setminus L$, thus we need not consider the values of φ on the set $\{p_1, p_2\}$. We also need not consider the values of φ on the set $\{p_3, p_4\}$, since either p_4 is not in L', or $L' = \{p_2, p_4\}$ in which case p_4 is in $L' \setminus L$ and p_3 is in $L \setminus L'$.

This leaves the two-element sets $\{p_1, p_3\}, \{p_1, p_4\}, \{p_2, p_3\}$ and $\{p_2, p_4\}$. Then, since each of these sets contains elements from two distinct connected components of H^-, we have that φ is a polymorphism, since by definition φ is ether a projection or a conservative majority operator on tuples from these sets. □

This completes the proof of Theorem 4.7.

4.3 Removing isolated strong loops

In this section we prove a generalization of Theorem 4.7, in which we allow H to have isolated strong loops. Recall that, a strong loop is isolated if it is not adjacent to any other strong loops. The main result of this section is the following theorem.

Theorem 4.11 *Let H be a trigraph path such that every strong loop is isolated and let H^- be the associated digraph of H. The list H-colouring problem is polynomial if the list H^--colouring problem is polynomial; and is **NP**-complete otherwise.*

Let H be a trigraph path such that every strong loop is isolated. We will assume again that list H^--colouring is polynomial, and consider an instance of the list H-colouring problem with representatives. We are given a digraph G, a set of lists $L(G)$, a set of non-empty parts S and a set of representatives V_S. We will *completely ignore* any part which does not have a representative. Thus we assume without loss of generality that $V(H) = S$. We also assume that the lists $L(G)$ are arc-consistent. Recall that the vertices of H are given in a path ordering $\{1, 2, ..., m\}$ so that adjacent vertices are consecutive in the order. We will treat the vertices of H as integers and say that y *is to the left of* x if $y < x$, and y *is to the right of* x if $x < y$. Finally, for a set of vertices $X \subseteq V(G)$ and a subset $B \subseteq V(H)$, we adopt the shorthand $L(X) \subseteq B$ to indicate that for each $v \in X$ the list $L(v)$ is a subset of B.

Our first step is to prove a useful fact, which will be used many times in what follows. For an instance $G, L(G)$ of the list H-colouring problem, we say that the *pre-image of a vertex $x \in V(H)$ is completely determined* if there is a set $C \subseteq V(G)$ such that $v \in C$ implies $L(v) = \{x\}$ and $v \in V(G) \setminus C$ implies $x \notin L(v)$. Observe that this implies that, for any homomorphism $f : G \to H$ with respect to $L(G)$, the pre-image $f^{-1}(x)$ is exactly the set C. In this case, we can delete x and obtain an equivalent problem, as shown in the following lemma.

Lemma 4.12 *Let H be a trigraph with a specified vertex x, let G be a digraph with* arc-consistent *lists $L(G)$. If the pre-image of x is completely determined, then there is a homomorphism $G \to H$ with respect to $L(G)$ if and only if there is a homomorphism $(G - C) \to (H - x)$ with respect to $L(G)$.*

Proof. Since $G - C$ and $H - x$ are contained in G and H, there is a homomorphism $G \to H$ with respect to $L(G)$ only if there also a list homomorphism $(G-C) \to (H-x)$ with respect to $L(G)$. On the other hand, suppose $g : (G-C) \to (H-x)$ is a homomorphism with respect to $L(G)$. Then we can define a homomorphism $g' : G \to H$ with respect to $L(G)$ as follows. For each $u \in V(G) \setminus C$ let $g'(u) = g(u)$, and for each $u \in C$ let $g'(u) = x$. It suffices to show that for all $u \in C$ and $v \in V(G) \setminus C$ the mapping $g'|_{\{u,v\}}$ is a homomorphism with

respect to $L(G)$. This follows from the fact that the lists $L(G)$ are arc-consistent. Recalling the arc-consistency procedure for trigraphs as described in Section 2.4; if $uv \in E(G)$ then, from the definition of arc-consistency, we have $g'(v) \in L(v)$ only if there exists a vertex z in $L(u)$ such that $zg'(v) \in E(H)$, but we have $L(u) = \{x\}$, and thus $g'(u)g'(v) \in E(H)$ as required. A similar argument holds in the cases that $vu \in E(H)$, $uv \in N(H)$, or $vu \in N(H)$. Thus, g' is a homomorphism with respect to $L(G)$. □

The lemma is useful in the following way. Suppose H is a trigraph path such that each strong loop is isolated, and with an associated digraph H^- such that the list H^--colouring problem is polynomial. Suppose we can "guess" the pre-image of each isolated strong loop, i.e. transform the lists $L(G)$ into lists $L^*(G)$ in which the pre-image of each isolated strong loop is completely determined. We could then use Lemma 4.12 to delete all isolated strong loops to obtain a trigraph H' without strong loops, and a corresponding instance G', $L^*(G)$. Theorem 4.7 implies that list H'-colouring is solvable in polynomial time, which in turn shows that the list H-colouring problem is polynomial.

As an example of how we might "guess", suppose we have an isolated strong loop x whose neighbours are loopless. Then in this case we can apply Lemma 4.5 to "guess" a clique C which maps to x. In fact, as explained in the proof of Lemma 4.5, we can enumerate all possible cliques C in polynomial time by enumerating split partitions. However, this approach doesn't get us too far, since it only applies when the neighbours of x are not weak loops. We thus need to develop a more general line of attack.

A note about "guessing". We remark that, in general, it is not necessary to "guess" all possible cliques which could map to x under a homomorphism. It is sufficient to "guess" a clique C such that if there is a homomorphism $f : G \to H$ with respect to $L(G)$ then there is a homomorphism $f' : G \to H$ with respect to $L(G)$ for which $f'^{-1}(x) = C$. Thus, it is only necessary to enumerate a collection $\mathcal{C}$ of cliques which is guaranteed to contain such a C.

A *stable-pair* in G is a pair of disjoint sets (A, B) such that there are no edges between the vertices of A and B. For example, the clique-cutset problem can be phrased in these terms as the problem of finding a partition of G into three non-empty sets A, B, C such that (A, B) is a stable pair and C is a clique. Our problem of finding a list H-colouring of G can be phrased in a similar way. Suppose x is a strong loop of H with representative v_x, then an H-colouring $f : G \to H$ induces a partition of G into three disjoint sets A, B, C such that f takes A to the left of x, B to right of x and C to x. Since H is a trigraph path the

sets (A, B) are a stable-pair. Thus, we can view list H-colouring as something very similar to a clique-cutset problem. Note however, that in this case the set C is not necessarily a cutset since one or both of A and B could be empty.

The notion of a separating family has been useful for solving the list clique-cutset problem and related problems (cf. for instance [28, 31] and also [50]). A family of sets $\mathcal{X}$ *separates cliques and stable pairs* of G if for any partition of $V(G)$ into a clique C and stable-pair (A, B) there is some $X \in \mathcal{X}$ such that X contains C and is disjoint from at least one of A or B. The set X from the definition is a slight relaxation of a clique-cutset, it is required to contain a clique C but may also contain vertices from one of the sets in the stable-pair (A, B).

Since we have representatives, we are really only interested in cliques C which contain the representative v_x of some strong loop x. We will therefore refine our definition of a separating family a little. We say that a family of sets $\mathcal{X}$ *separates cliques and stable pairs of G with respect to a strong loop x* if the following condition is met. For any partition of $V(G)$ into a clique C containing v_x, and a stable-pair (A, B), there is some $X \in \mathcal{X}$ which contains C and is disjoint from at least one of A or B.

Lemma 4.13 *Let x be a strong loop and let $\mathcal{X}$ be a family of sets which separates cliques and stable pairs with respect to x. If $f : G \to H$ is a homomorphism with respect to the lists $L(G)$ then some $X \in \mathcal{X}$ contains $f^{-1}(x)$.*

Proof. Let $f : G \to H$ be a homomorphism with respect to $L(G)$. Let A be the subgraph mapping to the left of x, B the subgraph mapping to the right of X and let C the subgraph mapping to x. Then C is a clique containing v_x and (A, B) is a stable-pair. Thus some $X \in \mathcal{X}$ separates C from (A, B). □

The following lemma tells us that we can always find such a separating family in polynomial time.

Lemma 4.14 *Let x be an isolated strong loop. We can enumerate in polynomial time a family of sets $\mathcal{X} = \{X_1, ..., X_n\}$, with at most n members, which separates cliques and stable-pairs of G with respect to x. Further, each $X \in \mathcal{X}$ contains v_x and is contained in the closed neighbourhood of v_x.*

Proof. This follows more or less directly from a result of Tarjan in [55]. According to [28] we can use the approach of [55] to enumerate in polynomial time a collection $\mathcal{E} = \{E_1, ..., E_n\}$

of sets which separates all cliques C and all stable-pairs (A, B) given that A, B, C partition the graph. We apply this to the underlying undirected graph of G.

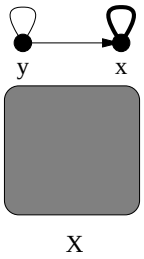
Now, we can define the family $\mathcal{X}$. For each E in $\mathcal{E}$ which contains v_x we form a set $X \in \mathcal{X}$ the restriction of E to the closed neighbourhood of v_x. It is easy to see that the family $\mathcal{X}$ satisfies the conditions of the lemma. □

Given a family $\mathcal{X}$ which separates cliques and stable-pairs with respect to x, we are assured that if there is a homomorphism of $f : G \to H$ with respect to $L(G)$ then some $X \in \mathcal{X}$ contains $f^{-1}(x)$. Our approach will be the following.

We consider each $X \in \mathcal{X}$ in turn. We remove x from the lists of all vertices not in X. If a list homomorphism exists with respect to $L(G)$ then it is a homomorphism with respect to at least one of these modified instances. Since each X is contained in the closed neighbourhood of the representative v_x then we can assume $L(X) \subseteq \{x-1, x, x+1\}$. For each X, we form two sub-problems; in the first problem we restrict $L(X)$ to $\{x-1, x\}$ (i.e. remove $x+1$ from all lists in X), in the second problem we restrict $L(X)$ to $\{x, x+1\}$. We claim that if a homomorphism exists with respect to this choice of X, it also exists with respect to at least one of these sub-problems. This is justified because, for any homomorphism $f : G \to H$ with respect to $L(G)$, there is a partition of $V(G)$ into sets A, B, C where the sets $A = \bigcup_{y<x} f^{-1}(y)$ and $B = \bigcup_{y>x} f^{-1}(y)$ are a stable-pair and $C = f^{-1}(x)$ is a clique containing v_x. Some set X separates C from (A, B), which implies that X is disjoint from either A or from B. Thus either $f^{-1}(x-1)$ or $f^{-1}(x+1)$ is disjoint from X.

For the remainder of this section we will fix a set X and assume that $L(X) \subseteq \{x, x-1\}$; the case $L(X) \subseteq \{x, x+1\}$ follows symmetrically. We will view our algorithm as an attempt to partition X into a set C mapping to x and a set $X \setminus C$ mapping to $x-1$. Our algorithm will modify the lists $L(G)$ to new lists $L^*(G)$ in which the list $L^*(v)$, for each $v \in X$, is either $\{x\}$ or $\{x-1\}$; the new lists $L^*(G)$ imply a partition of X into C and $X \setminus C$. The lists $L^*(G)$ have the property that the pre-image of x is completely determined. We show that if there is a list homomorphism of $G \to H$ with respect to $L(G)$ then there is a homomorphism of $G \to H$ with respect to $L^*(G)$, i.e. the lists $L^*(G)$ are reduced lists for $L(G)$. Thus we can apply the arc-consistency procedure for trigraphs described in Section 2.4 to $L^*(G)$ then use Lemma 4.12 to delete x from H. Clearly, we can apply this programme recursively to remove all strong loops from H.

We will start with a few cases for which partitioning of X is straightforward. For example consider the case that there is no symmetric edge between x and $x-1$.

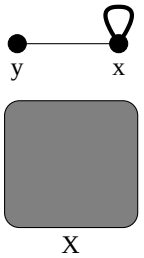


Lemma 4.15 *Let x be a strong loop, and let y be a vertex such that xy is not a symmetric edge. Let X be a set of vertices such that $L(X) \subseteq \{x, y\}$. If $f|_X : G \to H$ is a homomorphism with respect to $L(G)$ then $v \in f^{-1}(x)$ if and only if v is a symmetric neighbour of v_x.*

Proof. Let $f|_X : G \to H$ be a homomorphism with respect to $L(G)$, and consider $v \in X$. The lemma follows from a direct application of arc-consistency. First, observe that, since $f^{-1}(x)$ must induce a clique, we have $v \in f^{-1}(x)$ only if v is a symmetric neighbour of v_x. Now, if v is a symmetric neighbour of v_x, then $f(v) = x$ since at most one of xy, yx is an edge.□

Notice that the lemma completely determines the pre-image of x and so we can apply Lemma 4.12 to remove x.

Now consider the case that $x-1$ does not have a loop. In this case also there is a direct argument to partition X.



Lemma 4.16 *Let x be a strong loop and let y be an irreflexive neighbour of x. Let X be a set of vertices such that $L(X) \subseteq \{x, y\}$. There are only a linear number of homomorphisms of X to the graph induced in H by $\{x, y\}$, and these can be enumerated in polynomial time.*

Proof. Let $f : G \to H$ be a homomorphism with respect to $L(G)$. The pre-image of x and y is a split graph in G, thus f^{-1} induces a split graph on the set X. Recall from the proof of Lemma 4.5 that there are at most a linear number of split partitions of any graph, and these can be enumerated in polynomial time.□

Since there are only a linear number of homomorphisms, we can enumerate each in turn modifying the lists $L(G)$ to new lists $L^*(G)$ to reflect each. If there is a homomorphism of $f : G \to H$ with respect to $L(G)$, then f is a homomorphism with respect to at least one of the modified lists $L^*(G)$. Again, each set of lists $L^*(G)$ completely determines the pre-image of x, and we can apply Lemma 4.12 to remove x.

In the more general case where we cannot apply Lemmas 4.15 and 4.16, there are many possible partitions of the set X (in fact there can be exponentially many). Luckily in this case, by considering the rest of G, we can find a partition of X which is suitable, i.e. corresponds to a set of lists $L^*(G)$ which are reduced lists. Consider a connected component K of the graph $G - X$. Since H is a trigraph path it is easy to see that each of these components either maps entirely to the left of x, or entirely to the right. We distinguish two distinct types of components K:

- *grounded components*: We say that K is *grounded* if K contains a representative. Since, we assume the lists $L(G)$ are arc-consistent, we can assume that K contains only representatives to the left of x or to the right of x. In the first case we say that K is *grounded to the left of* x, and in the second that K is *grounded to the right of* x.

- *free components*: We say that K is a *free component* if $V(K)$ contains no representatives. For each free component K, it is possible that K maps to the left or right of x.

Figure 4.2 shows an example of the situation.

Lemma 4.17 *Let K be a free component of $G - X$. Let $f : G \to H$ be a homomorphism with respect to $L(G)$. Then the restriction $f|_{V(K)}$ is a homomorphism $K \to H^-$ with respect to $L(G)$.*

Proof. Let $f : G \to H$ be a homomorphism with respect to $L(G)$. We show that for any strong edge $yz \in E(H)$ and pair of vertices $u, v \in V(K)$ we cannot have $f(u)f(v) = yz$. Note that we admit the possibility the $y = z$. No vertex of K has x on its lists, thus we may assume that neither y nor z is equal x.

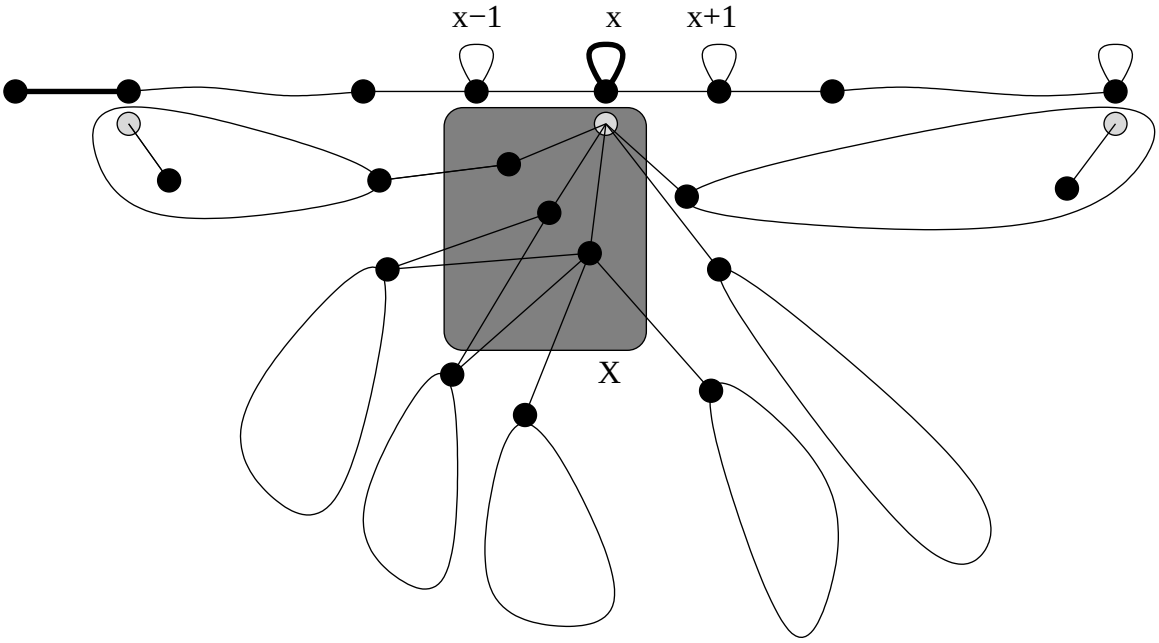


Figure 4.2: A trigraph path H is shown above, and an input digraph G is shown below. Grey vertices are representatives. An example of a set X with list $\{x-1, x\}$ is shown in the shaded box. Several connected components of $G-X$ are shown, some *grounded* (containing a representative) and some *free* (not containing any representative).

Since yz is a strong edge, each vertex in $f^{-1}(y)$ must be an in-neighbour of v_z. But this is impossible, since this implies u is adjacent with v_z, and so the representative v_z is in the free component K. Likewise, it is easy to see that $f(v) \neq z$.□

We present in Algorithm 1 a procedure, which finds a partition of X into a clique C which maps to x, and $X \setminus C$ which maps to $x-1$. We can then apply arc-consistency and delete x from H, and C from G to obtain an instance of the list $(H-x)$-colouring problem—just like in the cases described earlier. This algorithm can be applied recursively to delete all vertices with isolated strong loops from H.

The algorithm works as follows. We assume that the lists are initially arc-consistent. Notice that, by Lemmas 4.15 and 4.16, if there is not a symmetric edge between x and $x-1$ or, if $x-1$ is not a weak loop, then we already have a partition of X into C and $X \setminus C$. If our algorithm does not return this partition, then there is no list homomorphism. Thus we can assume that $x-1$ is a weak loop, and that there is a symmetric edge between x and $x-1$.

Initially, the clique C contains the vertices $v \in X$ such that $L(v) = \{x\}$. The clique C might be expanded by the algorithm, but never shrinks. We tacitly assume that all other vertices of X map to $x-1$. All free components start on the list $\mathcal{L}$: we assume that they all map to the left of x. In Step 5 the procedure TestLeft tests this assumption for each free component on the list $\mathcal{L}$. Each component can be tested independently, since from Lemma 4.17 we know that all free components map to weak edges of H. If a component fails this test then it must map to the right of x, and so we add it to the list $\mathcal{R}$. Once all free components have been tested, we return to the main algorithm.

Each component which is forced to the right can cause vertices to be added to the clique C; this is a simple consequence of arc-consistency. Thus, for each connected component K on the list $\mathcal{R}$ we need to update the lists $L^*(G)$ as follows. The vertices in K which are adjacent with a vertex in X must map to $x+1$. This is since $x+1$ is the only vertex to the right of x which is adjacent with a vertex in $\{x-1, x\}$. Similarly, the vertices in X which are adjacent with a vertex in K must map to x. Thus we can update the lists as follows, for $v \in K$ and $u \in X$ such that $uv \in E(G)$ let $L^*(v) = \{x+1\}$ and $L^*(u) = \{x\}$. We also update our set C to include the vertices which now have list $\{x\}$. We then apply the arc-consistency procedure for trigraphs from Section 2.4, to reflect the updated lists, and to verify that C is a clique. If any of the lists have changed, we need to return to Step 5 and retest all of the components on $\mathcal{L}$, with respect to the updated lists, and proceed as before.

Finally, once the lists no longer change in Step 7 we have identified the set C, so for each $w \in X \setminus C$ we restrict the list $L^*(w)$ to $\{x-1\}$. This is justified by the fact that vertices of $X \setminus C$ are not adjacent with vertices in $\mathcal{R}$. Since $x-1$ is a weak loop, and there is a symmetric edge between $x-1$ and x then this does not cause and problems. Since the lists changed we apply the arc-consistency procedure for trigraphs, to ensure that the lists $L^*(G)$ are arc-consistent. We can then return C and the updated lists $L^*(G)$.

The correctness of the algorithm will follow from the fact that at each stage, the lists $L^*(G)$ are reduced lists. If any list in $L^*(G)$ is empty, we can be certain that there is no homomorphism of G to H with respect to the original lists $L(G)$.

We now prove that Algorithm 1 is correct.

Lemma 4.18 *Let X be a separating set obtained by Lemma 4.14 and applying Lemmas 4.15 and 4.16. Then the lists $L^*(G)$ produced by Algorithm 1 are reduced lists.*

Input: An instance of the list H-colouring problem with representatives. A strong loop x, such that $x-1$ is not a strong loop. And a set X with $L(X) \subseteq \{x-1, x\}$ and such that $x \in L(v)$ only if $v \in X$.
Output: Reduced lists $L^*(G)$ and partition of the set X into two sets: $C = \{w \in X \mid L^*(w) = \{x\}\}$ and $X \setminus C = \{w \in X \mid L^*(w) = \{x-1\}\}$.

1 let $L^*(G) = L(G)$;
2 initialize $C := \{v \in X \mid L^*(v) = \{x\}\}$;
3 initialize $\mathcal{R} := \{$*grounded components to the right of* $x\}$;
4 initialize $\mathcal{L} := \{$ *the remaining connected components of* $G - X\}$;
5 run Algorithm 2:**TestLeft**(X, $\mathcal{L}$, $\mathcal{R}$) to update lists $\mathcal{L}$ and $\mathcal{R}$;
6 **for** *each* $K \in \mathcal{R}$ **do**
7 for each $v \in K$ adjacent a vertex in X set $L^*(v) := L^*(v) \cap \{x+1\}$;
8 for each $v \in X$ adjacent a vertex in K set $L^*(v) := L^*(v) \cap \{x\}$ and add v to C;
9 apply **Arc-consistency** to update lists $L^*(G)$;
10 if the lists changed goto step 5 (retest components in $\mathcal{L}$).
11 for $x \in X \setminus C$ set $L^*(w) := L^*(w) \cap \{x-1\}$;
12 apply **Arc-consistency** to update lists $L^*(G)$;
13 **return** C and $L^*(G)$.

Algorithm 1: Remove x

Input: A set X with $L(X) \subseteq \{x-1, x\}$ and such that $x \in L(v)$ only if $v \in X$. A set of free components $\mathcal{L}$.
Output: A set of components $\mathcal{L}$ which can map left of x, and a set of components $\mathcal{R}$ which cannot map left of x.

1 obtain $L^{**}(G)$ from $L(G)$ by removing all vertices to the right of x;
2 apply **Arc-consistency** to the lists $L^{**}(G)$;
3 **for** *For each* $K \in \mathcal{L}$ **do**
4 **if** K *is a free component and* $K \not\to H^-$ *with respect to* $L^{**}(G)$ **then**
5 move K from $\mathcal{L}$ to $\mathcal{R}$
6 **return** $\mathcal{L}$ *and* $\mathcal{R}$

Algorithm 2: TestLeft

Proof. We will show that at each stage of the algorithm the lists $L^*(G)$ are reduced lists. Steps 7 and 11 are the only steps where we explicitly alter the lists, so we will justify that the lists produced in these steps are in fact reduced lists.

In step 7 we update the list of v by setting $L^*(v) := L^*(v) \cap \{x+1\}$, for all $v \in K$ which are adjacent with some vertex $c \in X$. We justify that this is a reduced list. The component K is in $\mathcal{R}$ which implies that either K is grounded to the right of x, or that K failed the test in step 5. In either case if f is a list H-colouring of G then K must map to the right of x. By a simple consistency argument, it is easy to see that $L^*(v) = L(v) \cap \{x+1\}$ are the only possible lists for v—since $L(X) \subseteq \{x-1, x\}$.

In step 11 for $w \in X \setminus C$ we set $L^*(w) = L^*(v) \cap \{x-1\}$. Since C contains all vertices $c \in X$ with lists $L^*(c) = \{x\}$, then either $L^*(w) = \{x-1\}$ already, or $L^*(w) = \{x-1, x\}$. In any case w is adjacent with only vertices in $\mathcal{L}$ or X. We can assume that $x-1$ is a weak loop, and there is a symmetric edge between $x-1$ and x (otherwise we have a partition of X already by Lemmas 4.15–4.16).

There are two cases to consider, either both edges between $x-1$ and x are weak, or there is a strong edge between $x-1$ and x. In the first case there is no problem, since the vertices in $X \setminus C$ are only adjacent with vertices in $\mathcal{L}$ and X, and so we can force all vertices of $X \setminus C$ to map to $\{x-1\}$ without causing conflicts. Finally if there is a strong edge between $x-1$ and x, then we also show that conflicts do not happen. Without loss of generality suppose $(x-1, x)$ is a strong edge. Suppose there exists $w \in X \setminus C$ and $c \in C$ such that $wc \notin E(G)$ and $x-1 \in L^*(w)$. Since $L^*(c) = \{x\}$, this contradicts that $L^*(G)$ is arc-consistent after step 9. □

We now gather together the pieces of the proof of Theorem 4.11. We consider the list H-colouring problem with representatives, we consider all possible choices of representatives. We begin by finding for each strong loop x, a family of sets $\mathcal{X}$ which separates cliques and stable-pairs with respect to x. For each set $X \in \mathcal{X}$ we consider two cases, in the first we modify the lists $L(G)$ so that $L(X) \subseteq \{x-1, x\}$ and in the second $L(X) \subseteq \{x, x+1\}$. Thus if H has k isolated strong loops, we consider $n^k 2^k$ sub-problems (n^k choices for a set X and, for each of these, 2^k sub-problems to consider). Lemmas 4.13 and 4.14 and the discussion immediately following them justify that, there is a homomorphism $G \to H$ with respect to $L(G)$ if and only if there is a homomorphism $G \to H$ with respect to the modified lists of at least one sub-problems. We thus, focus on one such sub-problem for the remainder.

Consider some isolated strong loop x with separating set X and, without loss of generality, assume $L(X) \subseteq \{x-1, x\}$. Consider the effect of applying Algorithm 1. The lists $L^*(G)$ produced by Algorithm 1 are reduced lists. Thus, there is a homomorphism $G \to H$ with respect to the original lists $L(G)$ if and only if there is a homomorphism $G \to H$ with respect to the reduced lists $L^*(G)$. The lists $L^*(G)$ also have the property that the pre-image C of the isolated strong loop x is completely determined. Thus by Lemma 4.12, there is a homomorphism $G \to H$ with respect to $L^*(G)$ if and only if there is a homomorphism $(G - C) \to (H - \{x\})$ with respect to $L^*(G)$. Thus we can consider the list $(H - \{x\})$-colouring problem. Obviously, we can recursively apply Algorithm 1 to delete each isolated strong loop of H and obtain a trigraph H' without strong loops and a corresponding instance of $G', L(G')$ of the list H'-colouring problem. By repeated applications of Lemma 4.12, we then have that $G \to H$ with respect to $L(G)$ if and only if $G' \to H'$ with respect to $L(G')$. By Theorem 4.7 the list H'-colouring problem is polynomial if the list H^--colouring problem is polynomial, and is **NP**-complete otherwise. This implies that the list H-colouring problem is polynomial if the list H^--colouring problem is polynomial, and is **NP**-complete otherwise. This completes the proof of Theorem 4.11.

4.4 Removing bigger clusters of strong loops

In this section we show how to remove larger clusters of strong loops. We generalize the techniques used in the last section. Combined with results from earlier sections, this will complete the proof of Theorem 4.2.

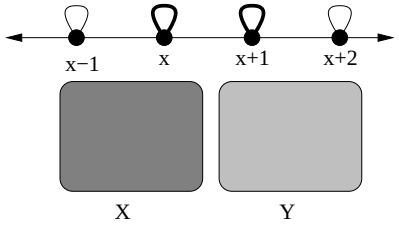
4.4.1 Removing clusters of exactly two consecutive strong loops

We will show how to remove groups of two consecutive non-isolated strong loops. Let x and $x+1$ be adjacent strong loops, and assume $x-1$ and $x+2$ are not strong loops. We will consider a family of sets $\mathcal{X}$ which separate cliques and stable-pairs with respect to x obtained from Lemma 4.14 and applying Lemmas 4.16–4.16, as well as a similar separating family $\mathcal{Y}$ with respect to $x+1$.

As in the proof of Theorem 4.11 we can generate two sub-problems for each $X \in \mathcal{X}$. In the first we let $L(X) \subseteq \{x-1, x\}$ and in the second $L(X) \subseteq \{x, x+1\}$. Likewise we can generate two sub-problems for each $Y \in \mathcal{Y}$, one in which $L(Y) \subseteq \{x, x+1\}$ and one in which $L(Y) \subseteq \{x+1, x+2\}$. Taking all possible combinations of these yields four sub-problems

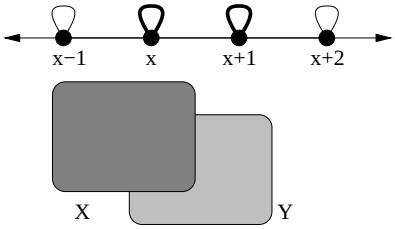
for each choice of X and Y. We will analyse these cases individually.

There are three cases to consider: $L(X)$ and $L(Y)$ are disjoint, $L(X)$ and $L(Y)$ have an overlap of one, or $L(X)$ and $L(Y)$ are identical. First consider that $L(X)$ and $L(Y)$ are disjoint, then $L(X) \subseteq \{x-1, x\}$ and $L(Y) \subseteq \{x+1, x+2\}$.



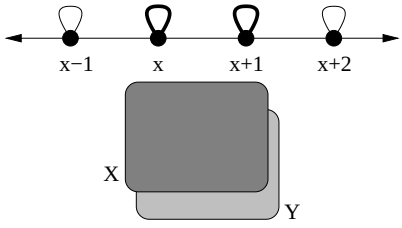
Since both $L(X)$ and $L(Y)$ contain at most one strong loop, we can simply apply Algorithm 1 to partition X and Y. Once X and Y are partitioned we have completely determined the pre-image of x and $x+1$ so we can apply Lemma 4.12 to delete x and $x+1$.

Next consider $|L(X) \cap L(Y)| = 1$; without loss of generality suppose that $L(X) \subseteq \{x-1, x\}$ and $L(Y) \subseteq \{x, x+1\}$.



We claim the set Y is uniquely partitioned by X. Indeed, consider $w \in Y \setminus X$, since $w \notin X$ then $x \notin L^*(w)$ implying that $L^*(w) \subseteq \{x+1\}$. On the other hand, observe that each $w \in Y \cap X$ has $L^*(w) \subseteq \{x\}$. Once Y is partitioned, we can apply Algorithm 1 to partition X. We can then apply Lemma 4.12 to delete x and $x+1$.

In the last case, we have $L(X)$ and $L(Y)$ both subsets of $\{x, x+1\}$.



All vertices with x on their list are in X, and all vertices with $x+1$ are in Y. Thus, if w is X or Y but not both then $L(w)$ can be reduced to either $L^*(w) \subseteq \{x\}$ or $L^*(w) \subseteq \{x+1\}$.

Thus consider $w \in X \cap Y$, we can assume without loss of generality that w is adjacent with a symmetric edge both of v_x and v_{x+1}. If a homomorphism exists, and there is a strong edge between x and $x+1$, then the set $X \cap Y$ is a symmetric clique. Otherwise we may assume that there is a symmetric edge between x and $x+1$ (else we could apply Lemma 4.15 to partition X and Y).

If there is a pair of vertices $w, w' \in X \cap Y$ such that w and w' are not symmetric neighbours, then w and w' must map to distinct vertices. This implies that the complement of the graph G_{XY} induced by the symmetric edges of $X \cap Y$ must be properly 2-colourable. Let G' be the complement of G_{XY}. Now, for each vertex v outside X, we consider the neighbours of v in $X \cap Y$ and identify them to a single vertex in G'. If G' is not 2-colourable after these identifications then there is no homomorphism. This is justified, since all neighbours of a vertex v must map to a vertex which is adjacent with $f(v)$. In particular if v is a vertex outside of X and Y, then the neighbours of v in $X \cap Y$ must have the same colour. So, for each connected component K of G' we arbitrarily choose a 2-colouring (by x and $x+1$) and apply arc-consistency to obtain reduced lists $L^*(G)$. We then test whether all free components of $G-(X \cup Y)$ which are adjacent with a vertex in K can map to H^- with respect to $L^*(G)$. If the test fails then we colour K with the other possible two colouring. We continue in this way until all components of G' are coloured. This is justified, since the 2-colourings of each connected component of G' are independent of each other and each free component is adjacent with at most one connected component of G'. If arc-consistency fails there is no homomorphism. Otherwise if arc-consistency succeeds we have a partition of X and Y into two symmetric cliques. Once we have this partition we proceed as before using Lemma 4.12 to delete x and $x+1$.

As in the case of isolated strong loops, we can apply this programme recursively to delete strong loops. Thus if H is a trigraph path with groups of at most two consecutive strong loops, we have the following theorem.

Theorem 4.19 *Let H be a trigraph path such that every strong loop is adjacent to at most one other strong loop and let H^- be the associated digraph of H. Then the list H-colouring problem is polynomial if the list H^--colouring problem is polynomial, and is **NP**-complete otherwise.*

4.4.2 Removing clusters of three or more consecutive strong loops

We now show how to handle a cluster of three or more strong loops. This will complete the proof of Theorem 4.2. The approach is similar to the cases described above. Consider a group of three or more consecutive strong loops. Let x be the leftmost strong loop in the cluster, and let y be the rightmost. Consider the family $\mathcal{X}$ of sets which separates cliques from stable-pairs with respect to x (from Lemma 4.14 applying Lemmas 4.16–4.16 if necessary) and a similar family $\mathcal{Y}$ of sets separating cliques and stable-pairs with respect to y. As before, there are four sub-problems to consider; however we can consider the cases for X and Y separately.

First, consider the case $L(X) \subseteq \{x-1, x\}$. Since $L(X)$ contains at most one strong loop, we can apply Algorithm 1 to partition X. The case for $L(Y) \subseteq \{y, y+1\}$ is the same. Now, consider the case that $L(X) \subseteq \{x, x+1\}$. In this case free components must go to left of x, since $x+2$ is a strong loop; the remaining components are grounded. We then apply the same line of reasoning we used to fix free components to the right in the case of isolated strong loops. For $w \in X$ adjacent with a vertex $v \in K$, where K is a component mapping to the left of x, we can set $L^*(w) = \{x\}$ and apply arc-consistency to update $L^*(G)$. This implies that $L^*(v) = \{x-1\}$. The case that $L(Y) \subseteq \{y-1, y\}$ follows by a similar argument.

In all four combinations of X and Y the lists $L^*(G)$ effectively partition the graph G into three parts. We have G_1 mapping completely to the left of the cluster, G_2 mapping inside the cluster, and G_3 mapping to the right of the cluster. Observe that the edges between G_1 and G_2 are fixed in place, i.e. if $uv \in E(G)$ with $u \in V(G_1)$ and $v \in V(G_2)$, then $L^*(u) = \{x-1\}$ and $L^*(v) = \{x\}$. A similar observation applies to edges between G_2 and G_3.

Assume we have applied arc-consistency for trigraphs to the lists $L^*(G)$. Let H' be the trigraph obtained from H by deleting all edges between $x-1$ and x and between $y+1$ and y. Let G' be the digraph obtained from G by deleting all edges between G_1, G_2 and G_3, and $L(G') = L^*(G)$. We claim that there is a list H'-colouring of G' with respect to $L(G')$ if and only if there is a list H-colouring of G with respect to $L^*(G)$. The proof of this fact is essentially the same as the proof of Lemma 4.12, and follows directly from the fact that the lists $L^*(G)$ are arc-consistent.

In effect, we have two disjoint list colouring problems, one for the cluster of strong loops and another for the remainder of H'. The list colouring problem for the cluster of strong loops is always polynomial. Since a vertex mapping to such a trigraph path must

be adjacent with the representative of a non-isolated strong loop we can apply Lemma 4.4 to obtain reduced lists of size at most two, which is easily solved by a reduction to **2SAT** (recall Lemma 2.7).

We can apply this programme recursively to obtain two disjoint list colouring problems: the list H^1-colouring problem, where H^1 is the sub-trigraph of H with no clusters of three or more strong loops, and the list H^2-colouring problem where H^2 is the sub-trigraph of H consisting only of clusters of three or more strong loops. It is easy to see that the list H-colouring problem is polynomial if the list H^1-colouring problem is polynomial, and is **NP**-complete otherwise.

We can now complete the proof of Theorem 4.2. Given a trigraph path H, with associated digraph H^-, we consider the list H-colouring problem. We consider all possible choices of representatives for H. We remove all groups of three or more consecutive strong loops, then all groups of two consecutive strong loops, then all isolated strong loops to obtain a trigraph H' without strong loops. We also generate a polynomial number of instances of the list H'-colouring problem such that there is a homomorphism $G \to H$ with respect to $L(G)$ if and only if there is a homomorphism $G' \to H'$ with respect to $L(G')$ for at least one of these instances $G', L(G')$. By Theorem 4.7, we have that the list H'-colouring problem is polynomial if the list H^--colouring problem is polynomial, and is **NP**-complete otherwise. Therefore the list H-colouring problem is polynomial if the list H^--colouring problem is polynomial, and is **NP**-complete otherwise. This concludes the proof.

4.5 Tractable trigraph paths

In this section we highlight some broad classes of trigraph paths with polynomial list H-colouring problems. This is accomplished by proving the associated digraphs H^- have polynomial list colouring problems. We will also point out the analogous results for matrix partitions.

Firstly, in the case of symmetric trigraph paths we have a complete characterization; the list homomorphism problem is polynomial only if the associated graph is bi-arc.

Lemma 4.20 *Let H be a symmetric trigraph path, with associated digraph H^-. The list H-colouring problem is polynomial if H^- is a bi-arc graph, and is **NP**-complete otherwise.*

Corollary 4.21 *Let M be a symmetric tridiagonal matrix, then the list M-partition problem*

*is polynomial if H^- is a bi-arc graph, and is **NP**-complete otherwise.*

When H is not necessarily symmetric we do not have a complete characterization, however we can highlight some classes with polynomial list colouring problems.

Suppose H is a trigraph path with no weak loops; then the underlying graph of each connected component of the associated digraph H^- is a loopless path. For brevity, we will refer to the connected components of H^- in this case as *semi-oriented paths (with no loops allowed)*. We show in the following lemma that H^- has a conservative majority polymorphism.

Lemma 4.22 *Let P be a semi-oriented path with no loops allowed, then P has a conservative majority polymorphism.*

Proof. We assume P is given in an ordering so that adjacent vertices are consecutive. We will treat the vertices of P as integers. We will construct a conservative majority polymorphism $g : P^3 \to P$.

Since P has no loops its underlying undirected graph is bipartite—let (P_1, P_2) be a bipartition of P. We define g as follows. If a, b, c are a triple such that a, b are in the same colour class and c is in the other, then $g|_{\{a,b,c\}}$ is essentially a binary operation on $\{a, b\}$ which is a projection onto its first co-ordinate—i.e. we ignore c and choose the first of a or b to appear. Otherwise, if a, b, c are all in P_1 or P_2, then $g|_{\{a,b,c\}}$ is the middle vertex with respect the ordering on $V(P)$. This definition guarantees that g is a majority operator, we verify that g is a polymorphism. Observe, that we can treat the two cases of the definition of g separately, since P is bipartite. In the first case, since g is essentially a binary projection it is easy to see that g is a polymorphism; thus we will focus on the second case.

Consider an edge $\vec{x}\vec{y} \in E(P^3)$, where $\vec{x} = (x_1, x_2, x_3)$ is a triple from P_1 and $\vec{y} = (y_1, y_2, y_3)$ is a triple from P_2. If x_1, x_2, x_3 are not all distinct and y_1, y_2, y_3 are not all distinct then, since g is a majority operator, $g(\vec{x})g(\vec{y})$ is one of the edges x_1y_1, x_2y_2 or x_3y_3. Thus, without loss of generality suppose x_1, x_2, x_3 are all distinct, and that $x_1 < x_2 < x_3$. If $g(y_1, y_2, y_3) = y_2$ then we are done. Thus, suppose that y_2 is not the middle vertex; without loss of generality suppose $g(y_1, y_2, y_3) = y_1$. Since x_1 is adjacent with y_1 we have $y_1 \leq x_1 + 1$; since x_2 is adjacent with y_2 we have $x_2 - 1 \leq y_2 \leq x_2 + 1$; and finally, since x_3 is adjacent with y_3 we have $x_3 - 1 \leq y_3$. But then, since $x_1 < x_2 < x_3$, we have $y_1 \leq x_2 - 1 \leq y_2 \leq x_2 + 1 \leq y_3$. Thus $g(\vec{x})g(\vec{y}) = x_2y_2$. □

Lemma 4.22 together with Theorem 4.2 implies the following result about trigraphs paths.

Theorem 4.23 *Let H be a trigraph path without weak loops, then the list H-colouring problem is polynomial.*

Theorem 4.24 *Let M be a tridiagonal matrix with diagonal over $\{0,1\}$ then the list M-partition problem is polynomial.*

Consider now a trigraph path H with no strong loops and such that the digraph induced by removing all loops from H^- contains no symmetric edges. That is, each connected component of the associated digraph H^- is an *oriented path with loops allowed*. Such paths have a conservative semi-lattice polymorphism.

Lemma 4.25 *Let P be an oriented path with loops allowed. Then P has a conservative semi-lattice polymorphism.*

Proof. We assume $V(P)$ is ordered so that adjacent vertices are consecutive, we will treat the vertices of P as integers. We define a conservative semi-lattice polymorphism $f : P^2 \to P$.

We define f as follows: $f(a,b) = f(b,a) = a$ if and only only if $a \leq b$. Now, consider an edge $\vec{x}\vec{y}$ in P^2, we may assume that x_1 and x_2 are distinct and that y_1 and y_2 are distinct (otherwise we are done), without loss of generality assume that $x_1 < x_2$. Suppose, for a contradiction, that $y_2 < y_1$. Since y_2 is adjacent to x_2, we have $x_2 - 1 \leq y_1$; since y_1 is adjacent with x_1, we have $y_1 \leq x_1 + 1$; and since $x_1 < x_2$ we have, $x_1 + 1 \leq x_2$. Thus, if $y_2 < y_1$ then we must have $y_2 = x_1$ and $y_1 = x_2$, but this is a contradiction, since there are no symmetric edges. Thus f is a polymorphism. □

Note that for oriented paths with loops allowed, there is also a conservative majority polymorphism, corresponding to taking the majority to be the "middle" vertex. The proof is essentially the same as in the case of the semi-lattice polymorphism demonstrated above.

Lemma 4.25 along with Theorem 4.2 implies the following result about trigraph paths.

Theorem 4.26 *Let H be a trigraph path, such that H^- is an oriented path with loops allowed; then the list H-colouring problem is polynomial.*

Theorem 4.27 *Let M be a tridiagonal matrix such that $M_{ij} \in \{1, \star\} \Rightarrow M_{ji} = 0$ then the list M-partition problem is polynomial.*

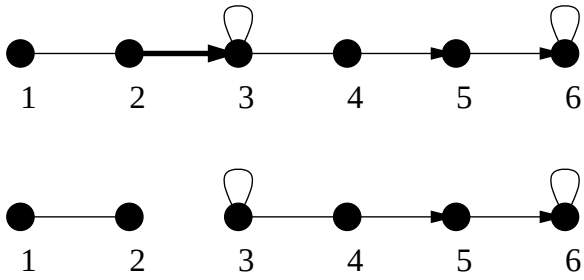


Figure 4.3: A trigraph path H is shown above, the associated digraph H^- is shown below. It is easy to show that H^- does *not* have a majority polymorphism, or a semi-lattice polymorphism. However, the connected component of H^- to the left has a majority polymorphism and the other has a semi-lattice polymorphism. Thus the list H-colouring problem is polynomial.

Finally, we note that there are other cases to consider, take for example the trigraph in Figure 4.3. It is provable that the associated digraph H^- does not have a majority polymorphism or semi-lattice polymorphism, however the list H^--colouring problem is polynomial since one connected component has majority and the other semi-lattice.

It would also be interesting to see if the results of Chapter 4 can be extended to other natural classes of trigraphs. For example, the class of trigraphs where the E relation induces a tree (with loops allowed) or the class of trigraphs where E induces a cycle (with loops allowed) seem like natural extensions of this work.

Appendix A

Trigraph List H-colouring Problems vs. c-CSP(H)

In this section we provide examples to demonstrate the difference between c-CSP and list trigraph H-colouring.

Perhaps some motivation is in order, we will first rephrase homomorphisms of digraphs to trigraphs in terms of homomorphisms between edge-coloured digraphs of the same type. Recall that trigraphs are a type of edge-coloured digraph with two binary relations, and some additional restrictions on the relations. Digraphs can also be seen as edge-coloured digraphs with two binary relations. Given a digraph G construct edge coloured digraph $G' = (V(G), E(G), E(\overline{G}))$. It is easy to verify that there is a homomorphism of the digraph G to the trigraph H if and only if there is a homomorphism of the edge-coloured digraphs G' to the edge-coloured digraph H. The list version of the problem can be modeled by introducing all possible unary relations into H.

Thus for a trigraph H, an instance of list H-colouring can be interpreted as an instance of c-CSP(H). Obviously the converse is not true, since not all edge-coloured digraphs with two binary relations correspond to digraphs. Thus the dichotomy result of Bulatov for c-CSP (Proposition 2.1) does not apply to the list H-colouring problem. We provide two examples here which demonstrate this difference.

First consider the trigraph shown on the left side of Figure A.1. As a trigraph homomorphism this is easily polynomial (it is a trigraph path with no weak loops). However, as a c-CSP problem this is **NP**-complete; since the N relation is the same as the digraph C_4^o

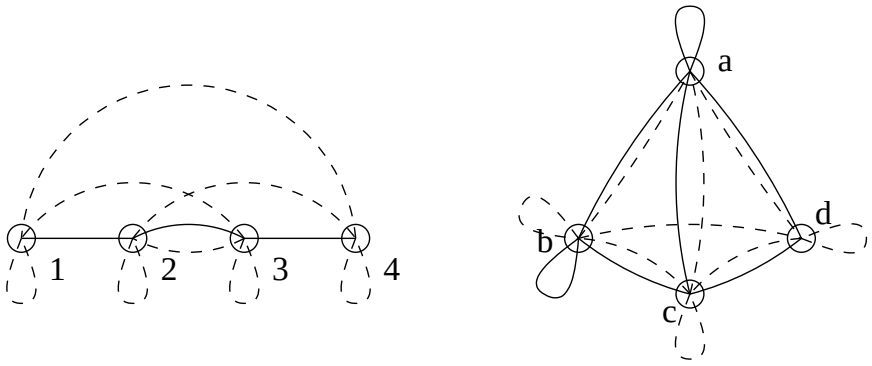


Figure A.1: Two trigraphs are shown for which the complexity of H-colouring and c-CSP(H) differ. For both the E relation is shown with solid lines and the N relation is shown with dashed lines. The trigraph on the right corresponds to the "stubborn" matrix.

which is **NP**-complete (cf. Figure 3.2).

Next consider the "stubborn problem" the corresponding trigraph H is shown on the right side of Figure A.1. The list H-partition problem is known to admit a quasi-polynomial algorithm. We will show that c-CSP(H) is **NP**-complete by applying the dichotomy result of Bulatov[7]. We will show that for the subset $\{c, d\}$ there is no polymorphism f such that $f|_{\{c,d\}}$ is a semi-lattice, affine, or majority operation. This will prove that list H-colouring is **NP**-complete. We can assume that all polymorphisms are conservative (because the presence of lists force this).

Consider first that $f|_{\{c,d\}}$ is a semi-lattice operator. To be a semi-lattice operator, on $\{c, d\}$, f must be a binary polymorphism which is associative, commutative, and idempotent on the set $\{c, d\}$. By commutativity we have that $f(c, d) = f(d, c)$. Consider the edges $cd, dc \in E$, applying f to these tuples yields $f(cd, dc) = ee$, where $e \in \{c, d\}$. But this is a contradiction since neither cc nor dd is a loop in E.

Now, consider that $f|_{\{c,d\}}$ is an affine operator. To be an affine operator, on $\{c, d\}$, f must be a ternary polymorphism such that $f(c, d, d) = f(d, c, d) = f(d, d, c) = c$. Consider the edges $da, cb, cb \in E$, applying f to these tuples yields

$$f\begin{pmatrix} d & c & c \end{pmatrix} = d \text{ and } \begin{pmatrix} d \\ a \end{pmatrix}, \begin{pmatrix} c \\ b \end{pmatrix}, \begin{pmatrix} c \\ b \end{pmatrix} \in E \Rightarrow f\begin{pmatrix} d & c & c \\ a & b & b \end{pmatrix} = \begin{pmatrix} d \\ e \end{pmatrix} \in E,$$

where $e \in \{a, b\}$. Since $db \notin E$ we have that $f(a, b, b) = a$. Now consider

$$f\begin{pmatrix} a & b & b \end{pmatrix} = a \text{ and } \begin{pmatrix} a \\ c \end{pmatrix}, \begin{pmatrix} b \\ a \end{pmatrix}, \begin{pmatrix} b \\ a \end{pmatrix} \in N \Rightarrow f\begin{pmatrix} a & b & b \\ c & a & a \end{pmatrix} = \begin{pmatrix} a \\ e \end{pmatrix} \in N,$$

where $e \in \{a, c\}$. Since $aa \notin N$ we have that $f(c, a, a) = c$. By symmetry of the affine operation we can permute arguments to get that $f(a, a, c) = c$. Finally we have arrived at a contradiction, consider the edges $ca, ac, ac \in E$, applying f to these tuples we must get $f(ca, ac, ac) = cc$ but $cc \notin E$.

So, consider $f|_{\{c,d\}}$ is a majority operator. To be a majority operator, on $\{c, d\}$, f must be a ternary polymorphism such that $f(c, c, d) = f(c, d, c) = f(d, c, c) = c$. We can use the majority to derive the value of $f(c, a, a)$ as follows:

$$f\begin{pmatrix} d & c & c \end{pmatrix} = c \text{ and } \begin{pmatrix} d \\ c \end{pmatrix}, \begin{pmatrix} c \\ a \end{pmatrix}, \begin{pmatrix} c \\ a \end{pmatrix} \in E \Rightarrow f\begin{pmatrix} d & c & c \\ c & a & a \end{pmatrix} = \begin{pmatrix} c \\ e \end{pmatrix} \in E$$

where $e \in \{a, c\}$. Since $cc \notin E$ we get that $f(c, a, a) = a$.

Now consider the value of $f(a, c, d)$. Let $f(a, c, d) = e \in \{a, c, d\}$. We can deduce using edges of E the following facts:

$$f\begin{pmatrix} a & a & a \end{pmatrix} = a \text{ and } \begin{pmatrix} a \\ a \end{pmatrix}, \begin{pmatrix} c \\ a \end{pmatrix}, \begin{pmatrix} d \\ a \end{pmatrix} \in E \Rightarrow f\begin{pmatrix} a & c & d \\ a & a & a \end{pmatrix} = \begin{pmatrix} e \\ a \end{pmatrix} \in E$$

$$f\begin{pmatrix} c & d & c \end{pmatrix} = c \text{ and } \begin{pmatrix} a \\ c \end{pmatrix}, \begin{pmatrix} c \\ d \end{pmatrix}, \begin{pmatrix} d \\ c \end{pmatrix} \in E \Rightarrow f\begin{pmatrix} a & c & d \\ c & d & c \end{pmatrix} = \begin{pmatrix} e \\ c \end{pmatrix} \in E$$

$$f\begin{pmatrix} d & d & c \end{pmatrix} = d \text{ and } \begin{pmatrix} a \\ d \end{pmatrix}, \begin{pmatrix} c \\ d \end{pmatrix}, \begin{pmatrix} d \\ c \end{pmatrix} \in E \Rightarrow f\begin{pmatrix} a & c & d \\ d & d & c \end{pmatrix} = \begin{pmatrix} e \\ d \end{pmatrix} \in E.$$

And so $f(a, c, d)$ is adjacent in E to all of a, c and d, thus $f(a, c, d)$ is not c or d (since neither are loops in E). Now using edges of N we derive:

$$f\begin{pmatrix} c & a & a \end{pmatrix} = a \text{ and } \begin{pmatrix} a \\ c \end{pmatrix}, \begin{pmatrix} c \\ a \end{pmatrix}, \begin{pmatrix} d \\ a \end{pmatrix} \in N \Rightarrow f\begin{pmatrix} a & c & d \\ c & a & a \end{pmatrix} = \begin{pmatrix} e \\ a \end{pmatrix} \in N.$$

And thus $f(a, c, d)$ must be adjacent a in N, thus $f(a, c, d)$ is not equal a (since a is not a loop of N). Thus there is no way to assign a value to $f(a, c, d)$ which is consistent with having $f|_{\{c,d\}}$ a majority operator. This completes the proof.

Bibliography

[1] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979. correction to algorithm appears in 14(4):195, 1982.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 14(4):195, 1982. errata.

[3] A. Brandstädt, P. L. Hammer, V. B. Le, and V. V. Lozin. Bisplit graphs. *Discrete Mathematics*, 299:11–32, 2005.

[4] R. C. Brewster, T. Feder, P. Hell, J. Huang, and G. MacGillivray. Near-unanimity functions and varieties of reflexive graphs. manuscript, 2008.

[5] A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM Journal on Computing*, 36(1):16–27, 2006.

[6] A. A. Bulatov. Mal'tsev constraints are tractable. Technical Report PRG-RR-02-05, Oxford University Computing Laboratory, 2002.

[7] A. A. Bulatov. Tractable conservative constraint satisfaction problems. Technical Report PRG-RR-03-01, Oxford University Computing Laboratory, 2002.

[8] A. A. Bulatov. The complexity of the conservative constraint satisfiability. *Doklady Mathematics*, 70(1):597–598, 2003.

[9] A. A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS,03)*, pages 321–330, 2003.

[10] K. Cameron, E. M. Eschen, C. T. Hoàng, and R. Sritharan. The list partition problem for graphs. In *Symposium on Discrete Algorithms*, pages 391–399. ACM/SIAM, 2004.

[11] K. Cameron, E. M. Eschen, C. T. Hoàng, and R. Sritharan. The complexity of the list partition problem for graphs. *SIAM Journal on Discrete Mathematics*, 21(4):900–929, 2007.

[12] M. Chudnovsky. Berge trigraphs. *Journal of Graph Theory*, 53:1–55, 2006.

[13] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164:51–229, 2006.

[14] S. Dantas, C. M. H. de Figueiredo, S. Gravier, and S. Klein. Finding H-partitions efficiently. *RAIRO–Theoretical Informatics and Applications*, 39:133–144, 2005.

[15] S. Dantas, C. M. H. de Figueiredo, S. Gravier, and S. Klein. Extended skew partition problem. *Discrete Mathematics*, 306:2438–2449, 2006.

[16] S. Dantas, C. M. H. de Figueiredo, S. Klein, S. Gravier, and B. A. Reed. Stable skew partition problem. *Discrete Applied Mathematics*, 143:17–22, 2004.

[17] C. M. H. de Figueiredo and S. Klein. The **NP**-completeness of multipartite cutset testing. *Congressus Numerantium*, 119:217–222, 1996.

[18] C. M. H. de Figueiredo, S. Klein, Y. Kohayakawa, and B. A. Reed. Finding skew partitions efficiently. *Journal of Algorithms*, 37:505–521, 2000.

[19] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55:87–101, 1992.

[20] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

[21] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory Series B*, 72:236–250, 1998.

[22] T. Feder and P. Hell. Full constraint satisfaction problems. *SIAM Journal of Computing*, 36:230–246, 2006.

[23] T. Feder and P. Hell. On realizations of point determining graphs, and obstructions to full homomorphisms. *DIscrete Mathematics*, 308(9):1639–1652, 2008.

[24] T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999.

[25] T. Feder, P. Hell, and J. Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003.

[26] T. Feder, P. Hell, and J. Huang. The structure of bi-arc trees. *Discrete Mathematics*, 307:393–401, 2007.

[27] T. Feder, P. Hell, S. Klein, and R. Motwani. Complexity of graph partitions. In *Thirty First Annual ACM Symposium on Theory of Computing*, pages 464–472. ACM, 1999.

[28] T. Feder, P. Hell, S. Klein, and R. Motwani. List partitions. *SIAM Journal on Discrete Mathematics*, 16(3):449–478, 2003.

[29] T. Feder, P. Hell, S. Klein, L. T. Nogueira, and F. Protti. List matrix partitions of chordal graphs. *Theoretical Computer Science*, 349:52–66, 2005.

[30] T. Feder, P. Hell, D. Král, and J. Sgall. Two algorithms for general list matrix partitions. In *Symposium on Discrete Algorithms*, pages 870–876. ACM/SIAM, 2005.

[31] T. Feder, P. Hell, and K. Tucker-Nally. Digraph matrix partitions and trigraph homomorphisms. *Discrete Applied Mathematics*, 154:2458–2469, 2006.

[32] T. Feder, P. Hell, and W. Xie. Matrix partitions with finitely many obstructions. *The Electronic Journal of Combinatorics*, 14(1):R58, 2007.

[33] T. Feder and M. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: a study through datalog and group theory. *SIAM Journal of Computing*, 28(1):57–104, 1998.

[34] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to NP-completeness*. W.H. Freeman, 1979.

[35] F. Gavril. Algorithms on clique separable graphs. *Discrete Mathematics*, 19(2):159–165, 1977.

[36] W. Gutjahr, E. Welzl, and G. Woeginger. Polynomial graph-colorings. *Discrete Applied Mathematics*, 35(1):29–45, 1992.

[37] P. L. Hammer and B. Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1980.

[38] P. Hell, S. Klein, L. T. Nogueira, and F. Protti. Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics*, 141:185–194, 2004.

[39] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory Series B*, 48:92–110, 1990.

[40] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*, volume 28 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, 2004.

[41] P. Hell, J. Nešetřil, and X. Zhu. Complexity of tree homomorphims. *Discrete Applied Mathematics*, 70:23–26, 1996.

[42] P. Hell, J. Nesetril, and X. Zhu. Duality and polynomial testing of tree homomorphisms. *Transactions of the American Mathematical Society*, 348(4):1281–1297, 1996.

[43] W. Imrich and H. Izbicki. Associative products of graphs. *Monatshefte für Mathematik*, 80:277–281, 1975.

[44] W. Imrich and S. Klavžar. *Product Graphs, structure and recognition*. Series in Discrete Mathematics and Optimization. Wiley-Interscience, 2000.

[45] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[46] P. G. Jeavons and M. C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79:327–339, 1995.

[47] R. M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, 1972.

[48] R. E. Ladner. On the structure of polynomial time reducibility. *Journal of ther ACM*, 22(1):155–171, 1975.

[49] L. Lovász. Operations with structures. *Acta Mathematica Academiae Scientiarum Hungaricae*, 18(3-4):321–328, 1967.

[50] L. Lovász. Communication complexity: A survey. In B. Korte, L. Lovász, H. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI Layout*, pages 235–265. Springer, 1990.

[51] R. M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37:93–147, 2003.

[52] D. J. Miller. The categorical product of graphs. *Canadian Journal of Mathematics*, 20(6):1511–1521, 1968.

[53] B. Reed. Skew partitions in perfect graphs. *Discrete Applied Mathematics*, 2007. DOI: 10.1016/j.dam.2007.05.054.

[54] M. Sipser. *Introduction to the theory of computation*. Thomson Course Technology, 2 edition, 2006.

[55] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55(2):221–232, 1985.

[56] W. Trotter and J. Moore. Characterization problems for graphs, partially ordered sets, lattices, and families of sets. *Discrete Mathematics*, 16(4):361–381, 1976.

[57] K. Tucker-Nally. List M-partitions of digraphs. Master's thesis, Simon Fraser University, 2003.

[58] P. M. Weichsel. The kronecker product of graphs. *Proceedings of the American Mathematical Society*, 13(1):47–52, 1962.

[59] S. Whitesides. An algorithm for finding clique cut-sets. *Information Processing Letters*, 12(1):31–32, 1981.