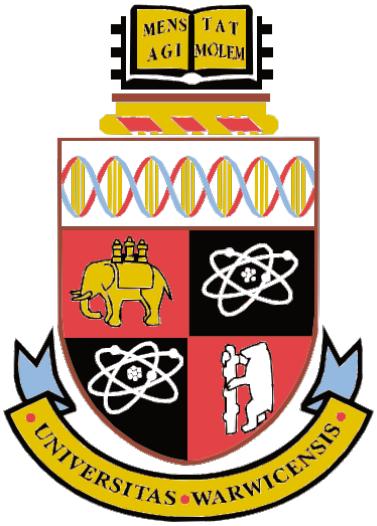


Distortion Colourings on Bipartite Graphs

by

Christopher Hickey

MA4K9 Dissertation

Submitted to The University of Warwick

Mathematics Institute

April, 2016

Contents

1 Introduction

Before going into the details of the problem we will be investigating in this report, let us first look into the motivation behind this problem (as set up in [2]). If we imagine we have two sets of computers, with each of these computers being a 'vertex', and we want to send messages from one set, the transmitters, to the other, the receivers.

We can represent these messages as 'edges' between the two sets of vertices and we can add a cost to these edges, $c(e)$. We will have multiple messages coming in to each of the receivers, however if we have the condition that a transmitter can not transmit and a receiver can not receive multiple messages at the same time, we have to figure out how to send the messages in such a way to avoid conflicts. We assign the transmitter end of each edge, e, a number, $f(e)$, corresponding to when we send the message, and then we have that the receiver end of that edge will receive that message at $f(e) + c(e)$. Our problem is now trying to make sure that for each vertex of the graph, the transmitters never have a conflict in the $f(e)$ values for each transmitter and the receivers never have a conflict in the $f(e) + c(e)$ values coming into each receiver.

In the actual definition of the problem which will we see in the next section, we restrict our possible values of $f(e)$ to n and add the condition that the receiving end gets assigned the number $f(e) + c(e)\, mod(n)$. The question we aim to answer is what the conditions on the minimum value of n is that will work for all graphs of particular types. The problem discussed here is an example of a 'delay colouring', where the 'colours' are the values at the end of the edges. Another way to think of the problem is where the edges, as oppose to adding a constant to the colour mod n, change the colours through permutations, this is a 'distortion colouring', and will be the primary focus of this report.

2 Definitions

To begin with, we will introduce the necessary definitions and concepts from graph theory required for this problem. In this section, the source of these definitions is [3].

2.1 Initial Definitions

We will consider two different types of graph in the report, simple bipartite graphs, and bipartite multigraphs. Before we define bipartite graph, we shall define the notion of a simple graph, and a multigraph.

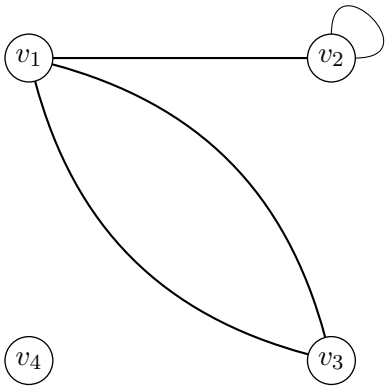


Figure 1: A multigraph, with
$V = \{v_1, v_2, v_3, v_4\}$
$E = \{v_1v_2, v_1v_3, v_1v_3, v_2v_2\}$

Definition 1. *A **simple graph** is a pair $G = (V, E)$ with $E \subseteq V^2$. The elements of V are called **vertices**, and elements of E are called **edges**. All edges in G can be represented as $e = uv$ for $u, v \in V$ with $u \neq v$.*

The fact that E is a subset of V^2 is vital here, this means we cannot have multiple edges between the same two vertices. This leads to the following extension of the definition.

Definition 2. *A **multigraph** is a pair $G = (V, E)$ with V the set of vertices, and E the **multiset** of edges, with all $e \in E$ a member of V^2. A multiset simply means a set where we are allowed repetition.*

This is essentially saying that a multigraph is a graph where we are allowed multiple edges between vertices, and edges that loop back onto the vertex they originate from.

We'll next define some important terminology within graph theory.

Definition 3. *These words help describe relationships between edges and vertices within graphs:*

> ***Incident*** *We say an edge e is incident to vertex v if v is at one of the ends of e.*
>
> ***Adjacent*** *Two vertices are adjacent if there is an edge between them.*
>
> ***Independent*** *Two vertices are independent if they are not adjacent.*
>
> ***Path*** *A path is a sequence of adjacent vertices.*
>
> ***Connected*** *A graph is connected if every vertex is attached to every other vertex by a path.*

With this, we can finish off our last few definitions we require.

Definition 4. *The **degree of a vertex** $v \in V$, denoted $deg(v)$, is the number of edges incident to v. The **degree of a graph** G, $deg(G)$, is $|V|$.*

In Figure 1 the vertices v_1 and v_2 have degree 3, $deg(v_3) = 2$, $deg(v_4) = 0$ and $deg(G) = 4$.

The final general graph theory definition we require is the following method to represent a graph which can provide insight into how the graph is structured, without having to draw out the entire graph.

Definition 5. *If we have a graph $G = (V, E)$ with $deg(G) = n$, the **adjacency matrix**, M_G, of G is a n by n matrix with $(M_G)_{ij}$ = number of edges in E between v_i and v_j.*

The adjacency matrix of the graph in Figure 1 is:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Note that the sum of the elements in M_G is twice the number of edges, because each edge is counted twice (once for each end). In the case of simple graphs, each element of the adjacency matrix will be 0 or 1, and there will be no non-zero elements on the diagonal (as diagonal entries are non-zero if and only if there are loops).
With these definitions we can now move on to the specific class of graphs we will be looking at in this report.

2.2 Bipartite Graphs

In the introduction we discussed a graph with two sets of vertices, and edges between these two sets, we will now define this concept properly.

Definition 6. *A **bipartite graph** is a graph, $G = (V, E)$ where V can be split into two distinct sets A and B, called a **bipartition**, where*

$$\forall a_i,\ a_j \in A,\ a_i \text{ and } a_j \text{ are independent}$$
$$\forall b_i,\ b_j \in B,\ b_i \text{ and } b_j \text{ are independent}$$

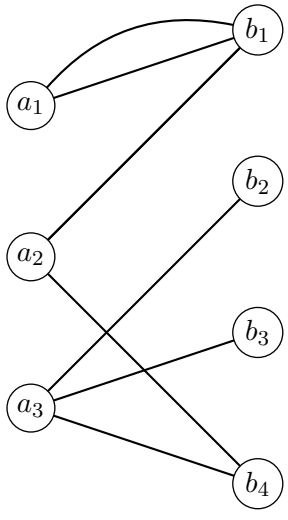


Figure 2: A Bipartite Graph

This is to say that every edge of G has one endpoint in A, and one endpoint in B. If we look at the adjacency matrix of a bipartite graph, we can see a clear pattern. For example, the adjacency matrix of Figure 2 is as follows.

$$\begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} a_1 & a_2 & a_3 & b_1 & b_2 & b_3 & b_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

As you can see, as A and B are both sets of independent vertices (called *independent sets*), we get that the adjacency matrix will be a block matrix of the form

$$\begin{pmatrix} 0_{|A|} & C \\ C^T & 0_{|B|} \end{pmatrix}$$

Where C is a $|A|$ by $|B|$ matrix. C is a very useful matrix, as we will see later on in the report.

Definition 7. *A **biadjacency matrix** of a bipartite graph is the $|A|$ by $|B|$ from the top right of the adjacency matrix, as in the diagram above.*

Another important concept within bipartite graphs are the *matchings*.

Definition 8. *A **matching** of a bipartite graph is a set of pair-wise non-adjacent edges. A matching is a **maximum matching** if it contains the maximum number of edges possible for that graph. A matching is a **perfect matching** which involves every vertex of G.*

We will see that matchings are closely related to the concept of a distortion colouring.

2.3 Graph Colourings

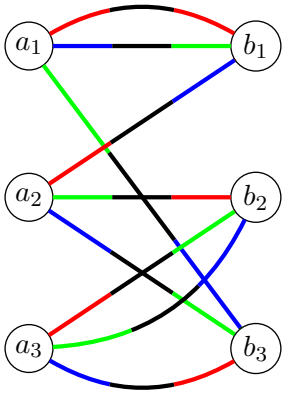
A popular topic in graph theory is how to colour a graph. When we talk about colouring a graph, we usually are referring to labelling the vertices or edges with a number, which can represent a colour. There are many different ways a graph can be coloured. One of the most common graph colourings is *Proper Vertex Colouring*, which is when we assign each vertex a colour such that no two vertices with the same colours are adjacent. Another type of colouring is *Proper Edge Colouring*, where we colour each edge such that no vertex has two edges of the same colour incident to it.

2.3.1 Distortion Colouring

In distortion colourings, we have a bipartite graph $G = (\{A, B\}, E)$, and a set of colours we can use, $Col = \{1, ..., n\}$. Each edge, $e = ab$ (with $a \in A, b \in B$), in our graph has a bijection $r_e : Col \to Col$ associated to it. For each edge $e = ab$ we assign some colour, $c \in Col$, to the A-end of e, and then assign $r_e(c)$ to the B-end of e. A distortion colouring of $G = (\{A, B\}, E)$ with $r_e \, \forall e \in E$ can be considered as a function $f : E \to Col$ where we have the A-end of each edge, e, being coloured $f(e)$, and the B-end of e being coloured $r_e(f(e))$. We call f a *n-proper distortion colouring* if

$$\forall a \in A, \ \forall e, e' \in E, \ e \neq e', \ \text{incident to } a, \ f(e) \neq f(e')$$
$$\forall b \in B, \ \forall e, e' \in E, \ e \neq e', \ \text{incident to } b, \ r_e(f(e)) \neq r_{e'}(f(e'))$$

The following figure demonstrates a distortion colouring.



Edge	0	1	2
a_1b_1 Top	0	1	2
a_1b_1 Bottom	1	2	0
a_1b_3	1	0	2
a_2b_1	2	1	0
a_2b_2	0	2	1
a_2b_3	1	2	0
a_3b_2 Top	2	0	1
a_3b_2 Bottom	1	0	2
a_3b_3	2	1	0

Figure 3: A 3-proper distortion colouring of a bipartite graph. To demonstrate the colouring in this example, we colour the graph with blue where the edge is assigned 0, green for 1, and red for 2.

In the above diagram, each vertex in G has degree 3, and we're able to colour G with just 3 colours, we will see in the next section that this is not always the case.

2.3.2 Delay Colourings

A simpler case of distortion colourings are *Delay Colourings*. In this case, we still have a bipartite graph $G = (\{A, B\}, E)$, and a set of colours we can use, $Col = \{1, ..., k\}$ but each edge, $e = ab$, has a cost, $c(e) \in \mathbb{Z}$. A delay colouring is a way to colour the graph, $f : E \to Col$, where $f(e)$ is the colour of the A-side of e, but the colour of the B-side of e

is $f(e) + c(e) \, mod \, k$. A *proper k-delay colouring* is a colouring, f, where

$$\forall a \in A, \ \forall \text{ distinct } e, e' \in E \text{ incident to } a, \ f(e) \neq f(e')$$
$$\forall b \in B, \ \forall \text{ distinct } e, e' \in E \text{ incident to } b, \ f(e) + c(e) \not\equiv f(e') + c(e') \, mod \, k$$

The final two subsections of the introduction will introduce concepts that we later use in section 4 to formulate alternative problems equivalent to the distortion colouring problem.

2.4 Directed Graphs

We shall look once more at some graph theory. All the graphs we have discussed so far have been *undirected*, that is to say, we have been dealing with the case where edges simply connect two vertices, and we can 'travel' in both directions around the graph. This isn't always the case.

Definition 9. *A **directed graph** is a graph, $G = (V, E)$, where each edge in G is an ordered pair (u, v) for $u, v \in V$, and the edge is considered to have a direction from u to v.*

By adding the concept of direction to graphs, a path between two vertices in G is now a more complex idea, we need to make sure the path goes the right way along each edge. Furthermore we can introduce *edge weightings* or *capacities*, for example in Figure 2.4 we see a directed graph where each edge has a weighting.

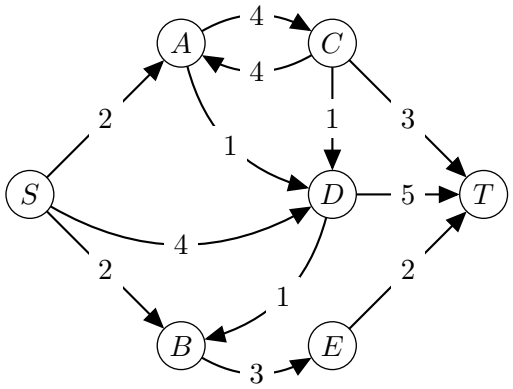


Figure 4: A weighted directed graph.

When we have a directed graph with a weighting, we can begin to think about the set of problems called *Network Flow Problems*. In a network flow problem, we can think of the edges as pipes, and the weights of the edges as the maximum capacity of the pipes. We consider paths between vertices in G, these paths will have a 'weight', we define this weight to be the minimum edge weight along the path, as if we considered sending a flow along this path, we could only send as much as the lowest weighted edge could handle. For example, for Figure 2.4 we can consider paths between S and T, there is a path of

weight 2, S-A-C-T. Network flow problems are of major interest within graph theory, and the problems involved have been studied intensely. We will see later on that our distortion colouring problem can be formulated as a network flow problem.

2.5 Matrices

Before we get on to the results involving distortion and delay colourings, let us first define some matrices which will be important in Section 4. Firstly, we define the following:

Definition 10. *The* ***standard basis of*** $\mathbb{R}^n$ *is the set of* n *vectors* $e_1, ...e_n$ *where for* $k \in 1, .., n$ *the* i*th element of* e_k *is*

$$(e_k)_i = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases}$$

This allows us to easily define a matrix that represents a permutation (distortion).

Definition 11. *A* ***permutation matrix*** *is a* $n \times n$ *matrix, M, where each row of M is a different vector from the standard basis of* $\mathbb{R}^n$

If we had a permutation, P, of a set of 4 elements, $\{1, 2, 3, 4\}$, written as $(1, 3, 2, 4)$, then we can write the permutation matrix of P as $M_P = (e_1, e_3, e_2, e_4)$. It follows that M_P multiplied by the 4×1 vector $(1, 2, 3, 4)^T$ will give us the permutation P. Distortions in distortion colourings can be represented by these matrices, and we will see why this is a useful idea in Section 4.

3 Current Results

My initial reading material was a paper by my supervisor, Agelos Georgakopoulos. In this paper, Delay Colourings of Cubic Graphs [1], he proves the following statement.

Theorem 1. *For every bipartite multigraph G of maximum vertex degree 3, and with any edge distortions of $\{1, 2, 3, 4\}$, there is a 4-proper distortion colouring of G.*

In proving the theorem he describes a method to find the 4-proper distortion colouring, however, this method will not work for bipartite graphs with maximum vertex degree greater than 3. His method used the fact that we can always add 'dummy edges' to make any graph with maximum vertex degree 3 3-regular (meaning every vertex has degree 3), and then he uses that any 3-regular bipartite graph has 3 perfect matchings, and that if we combine two of these we can get a 2-regular bipartite graph which we can use to build a colouring. The method works because a 2-regular graph is a cycle, which a path that start and ends at the same vertex, and this is why we can not extend this to graphs with maximum vertex degree greater then 3, as these will not be able to be made into a single matching and a cycle. A few years before this paper, Noga Alon and Vera Asodi wrote a paper 'Edge Coloring with Delays' [2], in which they proved that:

Theorem 2. *Every simple bipartite graph with maximum vertex degree D and any edge delays can be $(1 + o(1))D$-proper delay coloured.*

That is to say, eventually you will be able to D+1-proper distortion colour a simple bipartite graph, as the $(1 + o(1))D$ tends to $1 + D$ as D increases. They proved this using probabalistic methods, and did not find an algorithm that could provide a colouring. They also discussed a counterexample to the idea that every simple bipartite graph with maximum vertex degree D and any edge distortions on $\{1, 2, 3, ..., D\}$ has a D-proper distortion colouring, by considering the case when every distortion except one was the identity distortion. Both of these papers contain conjectures regarding the case where the graph has any vertex degree. These are

Conjecture 3. *Every bipartite multigraph with any edge distortions and maximum vertex degree D has a $D+1$ proper distortion colouring.*

Conjecture 4. *Every simple bipartite graph with any edge distortions and maximum vertex degree D has a $D+1$ proper distortion colouring.*

Conjecture 4 follows from Conjecture 3, and it is mostly Conjecture 4 that I will be looking into.

4 My Results

4.1 Graphs Where A and B Have Different Maximum Vertex Degree

An immediate question that arises from Theorem 1 is what happens if we have the bipartite multigraph $G = (\{A, B\}, E)$, but with

$$\max_{a \in A}(deg(a)) = 4, \quad \max_{b \in B}(deg(b)) = 3$$

Indeed it could be the case that we can colour G with any distortions on $\{1, 2, 3, 4\}$, as although we have this meaning every vertex in A has every possible colour edge incident to it, we have the freedom on the B side. The first investigations I did in this report we testing the following conjecture

Conjecture 5. *If we have a bipartite multigraph $G = (\{A, B\}, E)$ with $\max_{a \in A}(deg(a)) = 4$, $\max_{b \in B}(deg(b)) = 3$, then for any distortions of $\{1, 2, 3, 4\}$ on the edges of G, we can find a 4-proper distortion colouring of G.*

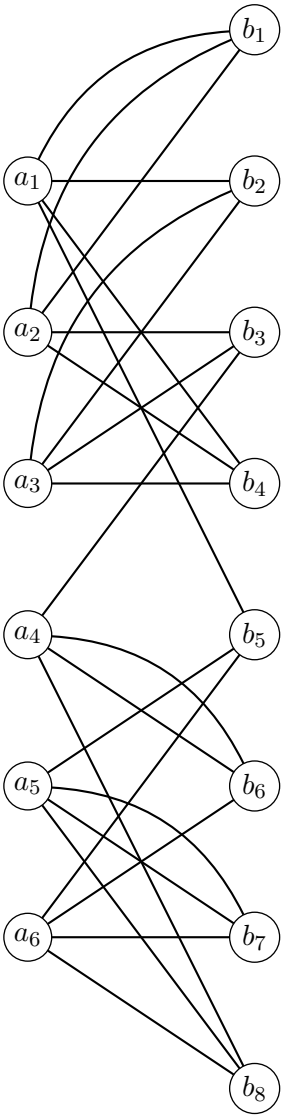
However, unfortunately this conjecture is not true. I wrote a MATLAB program which simply tried every possible distortion colouring f of G, in an attempt to find a counterexample, and it successfully found the counterexample in Figure 4. I initially had the program searching for graphs with $|A| = 4$, $|B| = 3$, but the program did not find a counterexample in 12 hours of being run, so I shifted the conditions to $|A| = 8$ and $|B| = 6$, and it found a counterexample within an hour. After this, I began to look into simple bipartite graphs, and what it means to have a distortion colouring for them.

4.2 Alternative Ways to Present the Problem

In order to tackle the problem, we can try to think of equivalent problems, which might be easier to solve. To construct these equivalent problems, the first concept we will introduce is what I call the *expanded distortion bipartite graph.*

Definition 12. *The **expanded distortion bipartite graph** of a bipartite graph $G = (\{A, B\}, E)$ with distortions on $\{1, ..., n\}$ where for each vertex of G in A, e.g. $a_i \in A$ we replace it with n new vertices, $a_{i1}, ..., a_{in}$ representing the colours that incident edges to a_i could have at the A-end, similarly, for each $b_j \in B$ we create $b_{j1}, ..., b_{jn}$ representing the colours that incident edges to b_j could have at the B-end. We then add for every $e = a_i b_j \in E$ n edges connecting a_{ix} to $b_{jr_e(x)}$ for $x \in \{1, ..., n\}$.*

With the expanded distortion bipartite graph, we can think of the problem now as a matching problem. In figure 6 a 3-proper distortion colouring would be a matching on the expanded distortion bipartite graph, however it's not quite as simple as any old matching. We require that the matching has the condition that for each $e = a_i b_j \in E$ from the



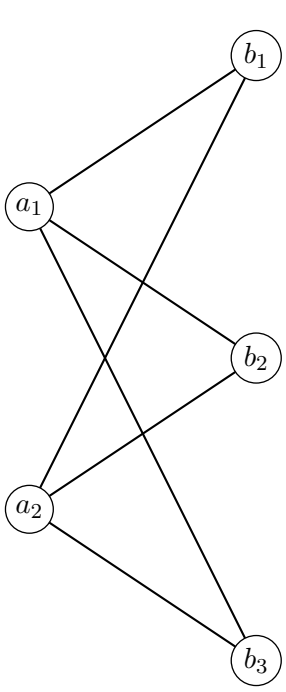
Edge	0	1	2	3
a_1b_1	1	3	2	0
a_1b_2	0	3	1	2
a_1b_4	1	0	2	3
a_1b_5	0	1	3	2
a_2b_1 Top	2	1	3	0
a_2b_1 Bottom	2	1	3	0
a_2b_3	1	2	0	3
a_2b_4	2	0	1	3
a_3b_2 Top	1	0	2	3
a_3b_2 Bottom	1	0	2	3
a_3b_3	0	1	3	2
a_3b_4	0	1	2	3
a_4b_3	3	0	1	2
a_4b_6 Top	3	1	0	2
a_4b_6 Bottom	3	1	0	2
a_4b_8	1	0	2	3
a_5b_5	1	0	3	2
a_5b_7 Top	1	2	0	3
a_5b_7 Bottom	1	2	0	3
a_5b_8	3	2	0	1
a_6b_5	1	2	0	3
a_6b_6	2	0	3	1
a_6b_7	2	0	1	3
a_6b_8	2	0	3	1

Figure 5: The counterexample to Conjecture 5.

original bipartite graph, the matching of the expanded distortion bipartite graph has just one edge from a_{ix} to $a_{ir_e(x)}$ for $x \in \{1, 2, ..., n\}$. For Figure 6 we could choose the matching

$$\{a_{10}b_{10}, a_{11}b_{21}, a_{12}b_{31}, a_{20}b_{12}, a_{21}b_{20}, a_{22}b_{32}\}$$

Trying to find a matching of the expanded distortion bipartite graph could be considered slightly easier than just colouring the original graph and trying to find a way to make the distortion colouring work, but essentially, it's no better. However, there is use in the expanded distortion bipartite graph. We will now examine two different methods of



Edge	0	1	2
a_1b_1	0	1	2
a_1b_2	2	1	0
a_1b_3	0	2	1
a_2b_1	2	0	1
a_2b_2	1	0	2
a_2b_3	0	1	2

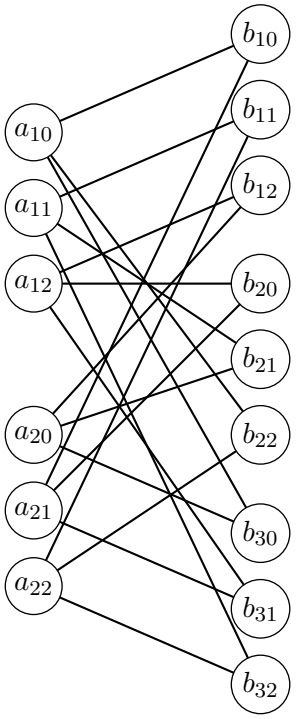


Figure 6: A bipartite graph, G; A table of distortions of {0,1,2} for the edges of G; The expanded distortion graph of G for these distortions.

presenting our problem for a simple bipartite graph with distortions. The first method involves using associating this problem to a class of already existing problems, network flow problems, as mentioned in Section 2.4. The second method is arguably simpler, and I will go into using this representation to provide a method of solving the problem in Section 4.3, however we will see that it does have issues that I have been unable to solve within this report.

4.2.1 Multi-Commodity Flow Problem

A specific kind of Network Flow Problem is the *Multi-Commodity Flow Problem*. In Section 2.4, we looked into simple network flows from one vertex, the source, to another vertex, the sink, and we can visualise this problem as trying to maximise the flow of water through

pipes connecting the source to the sink. The multi-commidity flow problem introduces the idea of multiple sources creating different commodities that we need to move through the graph to multiple sinks. The way to visualise this is a road network, with factories (sources) each creating a commodity that needs to be sent through a road network to a sink for that commodity, which could be a warehouse for that commodity, for example. The official definition of the problem is as follows [4].

Definition 13. *Given a directed graph $G = (V, E)$ where each $e \in E$ has a weight $c(e) \geq 0$, and a set of commodities $\{K_1, ..., K_n\}$ with $K_i = (s_i, t_i, d_i)$ where s_i represents the source of that commodity, t_i represents the sink, and d_i represents the demand, i.e. the desired flow from s_i to t_i. We define the flow for the commodity K_i to be f_i, where $f_i(e)$ is the flow of that commodity on the edge e. We similarly define $f(e) = \sum_{i=1}^{n} f_i(e)$, this is called the aggregrate flow of an edge e. The **Multi-Commodity Flow Problem** is the problem of finding a solution to the flow through G that satisfies the demand for all K_i, with the aggregrate flow of each edge not exceeding the total capacity for that edge. We also define the **Integral Multi-Commodity Flow Problem**, where $d_i, f_i(e) \in \mathbb{Z}\ \forall i \in \{1, .., n\}$.*

Figure 7 shows how the distortion colouring problem relates to the integral multi-commodity flow problem using the example expanded distortion graph from Figure 6.

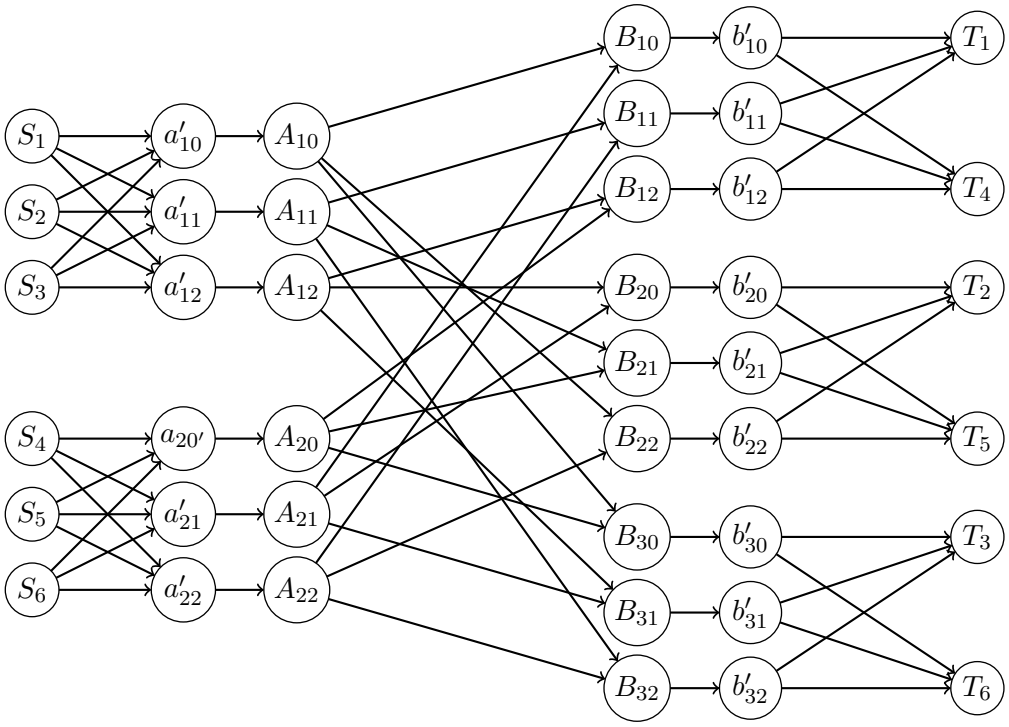


Figure 7: The formulation of the problem in Figure 6 as a multi-commodity flow problem. Here we have each edge having capacity 1, and the demand on each commodity being 1.

This graph may look complicated initially, but what is actually happening here is quite simple. Each commodity here relates to an edge of the original bipartite graph we are

trying to find a distortion colouring for. We formulate the k-proper distortion colouring of a bipartite graph $G = (V, E)$ with any distortions on {1,...,N} as the integral multi-commodity flow problem by first creating the expanded distortion graph, with edges of capacity 1, and the edges directed from the A-side to the B-side of the graph. We then adding a copy of each vertex in this graph, with each vertex connected to its copy by a edge of capacity 1, and the edges on the A-side directed from the copy to the original, and the edges on the B-side directed from the original to the copy. Finally, we add the sources and sinks by creating for each $e = a_i b_j \in E$ a commodity $K_m = (S_m, T_m, 1)$ where we attach each S_m to the copies of the $a_i x$ and T_m to the copies of $b_j x$ for all $x \in \{1, ..., N\}$.

This method of formulation can be seen in Figure 7, for example, we have S_1 representing the edge from a_1 to b_1. The reason this works is that this creates the matching we discussed earlier, as each path from S_m to T_m, representing the edge $e = a_i b_j$, can only take a route that takes it through the a_{ix} and b_{jy} vertices, where $x, y \in \{1, ..., N\}$, and we can not have the same colour being selected on a vertex of V as we have the copies, a'_{ix} and b'_{jy}, of a_{ix} and b_{jy} respectively having an edge of weight 1 between them, meaning only 1 commodity can flow through that particular colour of those vertices of G.

This formulation tells us that we can find a N-proper distortion colouring of a bipartite graph, G, with any edge distortions on $\{1, ..., N\}$ if we can find the desired multi-commodity flow in the directed graph representing G described above. However, the general integral multi-commodity flow problem is well known for being NP-complete, even with just 2 commodities [5]. This led me to believe that finding k-proper distortion colourings of bipartite graphs would be NP-complete, however I was unable to make much progress in a proof of this, so we will not go any further into it in this report.

4.2.2 Block Traversals

Whilst the multi-commodity flow problem formulation This expanded distortion graph will also have a biadjacency matrix. The biadjacency matrix of the expanded distortion graph in Figure 6 is shown in figure 8. This matrix will always have a particular structure. We will call matrices with this structure *block distortion matrices*, and we define these now.

Definition 14. *A $n \times m$* ***N,M-block D-distortion matrix*** *is a $nD \times mD$ block matrix made up of $D \times D$ distortion matrices with N non-empty blocks per row and M non-empty blocks per column (if $N = M$ we will only put N).*

The matrix in Figure 8 is a 2×3 3,2-Block 3-Distortion Matrix. From this definition we know define *block traversals*, and we will spend much of the rest of the project investigating the relationships between block traversals and block distortion matrices.

Definition 15. *A* ***block traversal*** *of a block distortion matrix, M, is a way of selecting*

a non-zero element from every non-zero block of M with the additional condition that we are selecting at most one element from each row and column of M.

The circled elements of Figure 8 are a block traversal.

	b_{10}	b_{11}	b_{12}	b_{20}	b_{21}	b_{22}	b_{30}	b_{31}	b_{32}
a_{10}	(1)	0	0	0	0	1	1	0	0
a_{11}	0	1	0	0	(1)	0	0	0	1
a_{12}	0	0	1	1	0	0	0	(1)	0
a_{20}	0	(1)	0	0	1	0	1	0	0
a_{21}	0	0	1	(1)	0	0	0	1	0
a_{22}	1	0	0	0	0	1	0	0	(1)

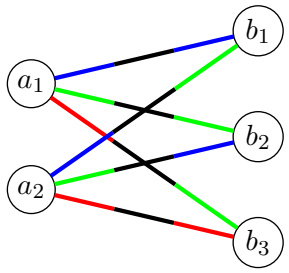


Figure 8: The graph on the right shows the distortion colouring that arises if we choose the circled elements. As earlier 0 is represented by blue, 1 by green, and 2 by red.

Every bipartite graph $G = (\{A, B\}, E)$ with maximum vertex degree N and $|A| = n$, $|B| = m$ and any distortions on $\{1, ..., D\}$ can be represented by an $n \times m$ N-block D-distortion matrix, as seen above, and in Figure 8 we see that a block traversal of the matrix gives rise to a 3-proper distortion colouring of the graph. As we show in this next theorem, it turns out this is always the case.

Theorem 6. *Given a simple bipartite graph G=({A,B},E) with edge distortions on D colours we can create a D-proper distortion colouring of G if and only if we can find a block traversal of the biadjacency matrix of the expanded distortion graph.*

Proof. This follows from the fact that when we have a expanded distortion graph, G_d, a proper distortion colouring of G would require us to have a matching of G_d such that if a_i and b_j are connected in G, then there has to be a single edge from a_{ix} to b_{jy}, $x, y \in \{1, ..., D\}$, in the expanded distortion graph. This is equivalent to having at least 1 element from each non-empty block of the biadjacency matrix of G_d, M_{G_d}, because if there is an edge between a_i and b_j in G, this edge will have a distortion, and so the respective block of M_{G_d} will be a permutation matrix. We need each chosen element from the blocks to be in different columns and rows because if we have two elements chosen from the same row of M_{G_d}, this would mean both chosen edges would share a vertex, which is not allowed in a matching. □

Thinking of the D-proper distortion colouring problem as this new problem of finding a block traversal of this block distortion matrix leads to some interesting results, for example if we consider complete bipartite graphs, we get the following result

Theorem 7. *Given a complete bipartite graph, $G = (\{A, B\}, E)$, where $|A| = |B| = D$, we can find a D-proper delay colouring if for each $a \in A$ (or for each $b \in B$) no two edges incident to it have the same delays.*

Proof. Assume each $a \in A$ (w.l.o.g could be each $b \in B$) has each edge with a different delay, then consider the biadjacency matrix of a expanded delay graph, for each column of block, pick the elements on the first row of each block from the first column, the elements on the second row of each block from the second column, and so on. As each $a \in A$ has a different delay, this means every column of blocks in the biadjacency matrix of a expanded delay graph has a different delay matrix, so if we choose all the elements on the first row of each block from the first column of blocks, we'll be guaranteed to be choosing at most one element from each column within this block column. Similarly for the second column of blocks, and so on. Therefore we can find a block traversal, and so we can find a D-proper delay colouring. □

In this proof, we are essentially building the biadjacency matrices from block traversals, as when we have a block traversal, this means we have chosen 1 element from this block, and so we can fill the rest of the block to make that 1 part of a delay matrix, and as we have a different column selected (as it's a block traversal) within each column of blocks, we get different delays within each block.
A potential method of proof for Conjecture 4 is if we consider two sets,

$X_{N,D}$ = Set of biadjacency matrices for expanded distortion graphs of a specific bipartite graphs of degree 2N with maximum vertex degree D and distortions on $\{1, ..., D+1\}$

$Y_{N,D}$ = Set of all possible block traversals of biadjacency matrices for expanded distortion graphs of a specific bipartite graphs of degree 2N with maximum vertex degree D and distortions on $\{1, ..., D+1\}$

It is easy to see that the size of $X_{N,D}$ is $(D+1)!^{ND}$, as we have ND edges in the graph, so ND non-zero blocks in biadjacency matrix, and we can pick any distortion of $\{1, ..., N+1\}$ to fill these blocks. Similarly, we can see the size of $Y_{N,D}$ will be $(D+1)!^{2N}$ as for each block column we have $(D+1)!$ ways to choose a different column per block, and for each block row, we have $(D+1)!$ ways to choose a different row per block, and we have $2N$ different block rows/columns. For each element in Y, we can make it an element of X by replacing the $D \times D$ zero matrix within each block containing a 1 with some $D \times D$ permutation matrix. We can make $(D+1)!^{2N} \times D!^{N^2}$ elements of X using this method, but of course some of these will be repeated, as each element of X could have multiple traversals (or none at all). I attempted to form a proof that the number of unique elements of X we can make in this way is the same as the size of X, however this proof failed due to the number of different traversals a element $x \in$ X has depends entirely on the distortions within x.

4.3 Using Block Traversals to Find Proper Distortion Colourings

In [1] Agelos provides a method of constructing a 4-proper distortion colouring of any bipartite multigraph with maximum vertex degree 3, but as mentioned in the Current Results section, this method does not extend to maximum vertex degrees higher than 3. I began looking into finding block traversals of D-Block D+1-distortion matrices, so as to provide a method to find D+1-proper distortion colourings on simple bipartite graphs of maximum vertex degree D. The following procedure works when $D = 3$, except for one case,however I believe this method has the potential to be expanded to work for $D = 3$ and to work for $D = 4$ (which we examine after this) and beyond. Before I go into the details of the method, I will first introduce the *Shortened $n \times m$ N,M-Block D-distortion matrices*, where we shorten the matrix by replacing each $D \times D$ block with a $1 \times D$ vector where the ith element of this vector is the row index of the non-zero element in the ith column of this block. For example, the shortened version of the 2×3 3,2-Block 3-Distortion Matrix from Figure 8 is

①	2	3	3	②	1	1	③	2
3	①	2	②	1	3	1	2	③

As you can see, a traversal of this matrix is now a way to choose a different column from each column of blocks (called a *Block Column*), and a different element from each row of blocks (called a *Block Row*).

4.3.1 In the Maximum Vertex Degree 3 Case

With this definition out the way, I can describe how my procedure works, with a running commentary as to what each step means (with two examples afterwards to demonstrate what is happening better):

Input Shortened 3-Block 4-Distortion Matrix, M, representing a connected bipartite graph (we can add this connected condition without loss of generality, if the graph is not connected, we can consider them as two graphs and run the procedure individually on each component)

Output Block Traversal of M (except in certain cases discussed below)

First, we go through each row of M and for each non-empty block (from here on in we shall simply refer to non-empty blocks as blocks, and empty blocks explicitly as empty blocks) choose 1 from the first block, 2 from the second and 3 from the third, then add an extra element at the end of the row, and set this element to be 4. This is the *Spare Colour*, because currently we are not using it for this row of the traversal.
Now we go through every block column of M from left to right, we aim to remove all the conflicts, which are columns which have multiple selected elements, whilst maintaining the

fact that each Block Row has a sequence of numbers with no repeats.
When we have a block column (we will refer to the block column we are removing conflicts from as β) with a conflict we try the following sequence:

Spare Colour Swap: For each block in β try swapping the selection from the chosen colour to the 'spare colour' in this block, if this removes the conflict, great! Update the spare colour column and move to the next block column. If this doesn't work, try the next method

NOTE If this method fails, this means that β must have two columns that contain both all the chosen colours and all the spare colours for their respective row. We call the colours that are in columns that have no selected elements yet 'legal colours'. In this case the 2 'legal colours' for each block in β have been chosen in a different block column.

Right Block Swap If β has a block column to the right of it that contains a block that is in the same block row as at least one block in β, then that block will have a colour, e.g. c', selected that is a legal colour for the block in β, where the current selected colour, c, is involved in a conflict. We can therefore change our selection in these blocks by swapping to choosing c' in the β block, and choosing c in the other block. This will make it so that we can use Spare Colour Swap to remove the conflict.

NOTE If this fails, we have a situation where the β has the last block for each of it's block rows. Therefore all block columns involving the two other blocks for each of these rows will be to the left of β. These block columns will have already had all conflicts removed, hence each block has 1 'legal colour'. We see we have the following cases:

Cases When We Look Left

1. If the legal colour for one of the involved blocks to the left is that rows spare colour, we can swap the legal colour for the spare colour, therefore allowing us to Spare Colour Swap in β and remove the conflict.

2. If the legal colour for one of the involved blocks to the right is the same as the chosen colour in the same block row within β we can switch to choosing the legal colour in that block and switch the choice in the respective block in β to the old colour selected in the block to the left.

3. The only remaining case is that the two blocks in the same block row as the blocks in β must have a legal colour that is the same as the selected colour in the other block. We now go through each row of β and try and perform Spare Colour Swaps on the other block columns with the same row involved to arrange the blocks in such a way that we can get to Case 1 or 2 and can remove the conflict in β. If this fails we move to the final possible situation.

NOTE At this point, we are in a specific case, we must have a situation like this:

①	2	3	4	1	②	3	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4
-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	-	-	-	-	-	-	-	-	4
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	4
3	2	①	4	1	3	②	4	-	-	-	-	-	-	-	-	-	-	-	-	1	2	③	4	4
3	2	4	①	-	-	-	-	1	3	②	4	-	-	-	-	-	-	-	-	1	2	4	③	4
-	-	-	-	-	-	-	-	2	3	4	①	1	3	②	4	2	1	4	③	-	-	-	-	4
-	-	-	-	3	2	4	①	-	-	-	-	1	2	4	③	1	2	③	4	-	-	-	-	4

As you can see in this diagram, we need to remove the conflict in the final block, and looking to the blocks to the left, we see we are in Case 3 as described above, and any attempt to use Spare colour Swaps fails, for example, if we look at the first block, we can try and use Spare Colour Swaps on the 4th and 5th row, but we see that this will simply result in the 1 and the 4 switching places, and doesn't seem to help us. If we are in the situation where we can't complete Case 3 for any row, we have the following circumstances

- Each block column with a block in the same row as a block of β must have it's two other blocks not sharing a row with a block of β. If this wasn't the case, we'd be able to use Spare Colour Swaps to change the selected elements in this block column and remove the conflict.
- These two blocks must have the selected colour of one block and the spare colour of the other block in the same column, as can be seen in the above diagram.

Unfortunately, this is where the procedure falls down. In this last case, we can of course swap the spare colour and the selected colour, and in all the cases I have looked at so far, this allows us to follow along the block rows of these columns and swap colours for spare colours in other block columns until finally coming across a way to swap the spare colour out in a block row of β, however, I have been unable to sum up these steps in such a way that I can guarantee the procedure will work every single time.

The majority of cases will be solved without coming across the problem in the last step. I will now go through some examples which show the procedure working.
To truly understand what is happening consider the following example. We wish to find a block traversal of the following shortened biadjacency matrix of a expanded distortion graph, with distortions on $\{1, 2, 3, 4\}$.

2	1	3	4	4	3	2	1	1	3	4	2	-	-	-	-	-	-	-	-
4	1	3	2	3	1	4	2	-	-	-	-	2	1	3	4	-	-	-	-
1	2	4	3	-	-	-	-	2	3	4	1	-	-	-	-	2	1	4	3
-	-	-	-	2	3	4	1	-	-	-	-	2	3	4	1	1	2	3	4
-	-	-	-	-	-	-	-	3	1	4	2	3	4	2	1	4	3	2	1

So, first we add the extra column and choose 1 to n in each row, and add the column of 4's to the end.

2	①	3	4	4	3	②	1	1	③	4	2	-	-	-	-	-	-	-	-	4
4	①	3	2	3	1	4	②	-	-	-	-	2	1	③	4	-	-	-	-	4
①	2	4	3	-	-	-	-	②	3	4	1	-	-	-	-	2	1	4	③	4
-	-	-	-	2	3	4	①	-	-	-	-	②	3	4	1	1	2	③	4	4
-	-	-	-	-	-	-	-	3	①	4	2	3	4	②	1	4	③	2	1	4

Next we inspect the first block, the first and second blocks have a conflict, we can do a Spare Colour Swap to swap the selection on the first row to 4 instead of 1, and then change the 'spare colour' for the first row to 1.

2	1	3	④	4	3	②	1	1	③	4	2	-	-	-	-	-	-	-	-	1
4	①	3	2	3	1	4	②	-	-	-	-	2	1	③	4	-	-	-	-	4
①	2	4	3	-	-	-	-	②	3	4	1	-	-	-	-	2	1	4	③	4
-	-	-	-	2	3	4	①	-	-	-	-	②	3	4	1	1	2	③	4	4
-	-	-	-	-	-	-	-	3	①	4	2	3	4	②	1	4	③	2	1	4

For the second block, we can see that the second and fourth row have a conflict, furthermore, we can't change either to the spare colour as this would form a conflict with the first row. So we attempt to do a Right Block Swap, and look to the fourth block. We can swap our choice of 1 in the fourth row of this block, with the choice of 2 in the fourth row of the fourth block to remove the conflict.

2	1	3	④	4	3	②	1	1	③	4	2	-	-	-	-	-	-	-	-	1
4	①	3	2	3	1	4	②	-	-	-	-	2	1	③	4	-	-	-	-	4
①	2	4	3	-	-	-	-	②	3	4	1	-	-	-	-	2	1	4	③	4
-	-	-	-	②	3	4	1	-	-	-	-	2	3	4	①	1	2	③	4	4
-	-	-	-	-	-	-	-	3	①	4	2	3	4	②	1	4	③	2	1	4

Next, the third block, this is an easy case, simply do a Spare Colour Swap to swap the fifth row of the block from the 1 to the 4, and change the spare colour from 4 to 1 for the fifth row.

2	1	3	④	4	3	②	1	1	③	4	2	-	-	-	-	-	-	-	-	1
4	①	3	2	3	1	4	②	-	-	-	-	2	1	③	4	-	-	-	-	4
①	2	4	3	-	-	-	-	②	3	4	1	-	-	-	-	2	1	4	③	4
-	-	-	-	②	3	4	1	-	-	-	-	2	3	4	①	1	2	③	4	4
-	-	-	-	-	-	-	-	3	1	④	2	3	4	②	1	4	③	2	1	1

In the fourth block, we have another situation where Spare Colour Swap will not work, hence we must look to the block to the right, the fifth block, and make some swaps, the legal colours in this case are 2 and 3, so we swap the chosen 1 in this block for the 3, and

change our selection of 3 to the 2 in the fifth block.

2	1	3	④	4	3	②	1	1	③	4	2	-	-	-	-	-	-	-	-	1
4	①	3	2	3	1	4	②	-	-	-	-	2	1	③	4	-	-	-	-	4
①	2	4	3	-	-	-	-	②	3	4	1	-	-	-	-	2	1	4	③	4
-	-	-	-	②	3	4	1	-	-	-	-	2	3	4	①	1	2	③	4	4
-	-	-	-	-	-	-	-	3	1	④	2	③	4	2	1	4	3	②	1	1

Now we only have to deal with the 5th block, however we can't do Spare Colour Swap, and we can't do Right Block Swap, as there are no matrices to the right of this block, so we must try checking the blocks to the left. We are in the first case as the third row of the first block can be swapped to it's spare colour. We can swap the 1 and 4 in the third row of the first block, and then swap the 3 to the 1 in the third row of the fifth block using Spare Colour Swap, and then use Spare Colour Swap again to swap the 3 to the 4 in the fourth row of this block.

2	1	3	④	4	3	②	1	1	③	4	2	-	-	-	-	-	-	-	-	1
4	①	3	2	3	1	4	②	-	-	-	-	2	1	③	4	-	-	-	-	4
1	2	④	3	-	-	-	-	②	3	4	1	-	-	-	-	2	①	4	3	3
-	-	-	-	②	3	4	1	-	-	-	-	2	3	4	①	1	2	3	④	3
-	-	-	-	-	-	-	-	3	1	④	2	③	4	2	1	4	3	②	1	1

And we are done, the circled elements will provide a block traversal of the biadjacency matrix of the expanded distortion graph, and hence find a 4-proper distortion colouring for the bipartite graph with these distortions represented here. The following example shows what happens in the 2nd case of looking left options.

2	3	4	①	1	3	4	②	-	-	-	-	-	-	-	-	2	③	1	4	4
-	-	-	-	-	-	-	-	①	4	3	2	②	3	4	1	1	4	2	③	4
-	-	-	-	2	①	4	3	3	②	4	1	-	-	-	-	1	③	2	4	4
3	①	4	2	1	4	②	3	-	-	-	-	1	4	③	2	-	-	-	-	4
2	4	①	3	-	-	-	-	3	4	②	1	2	③	4	1	-	-	-	-	4

We need to remove the conflict from the last column block, we see that we must look left, and we also see we are in case 3, if you look at the 3rd block column, you'll see we can swap the 1 for the 4 in the second row, and then the 2 for the 3 in the third row , and then swap the 3 and 2 in the third row of the fifth block, and therefore remove the conflict.

2	3	4	①	1	3	4	②	-	-	-	-	-	-	-	-	2	③	1	4	4
-	-	-	-	-	-	-	-	1	④	3	2	②	3	4	1	1	4	2	③	1
-	-	-	-	2	①	4	3	③	2	4	1	-	-	-	-	1	3	②	4	4
3	①	4	2	1	4	②	3	-	-	-	-	1	4	③	2	-	-	-	-	4
2	4	①	3	-	-	-	-	3	4	②	1	2	③	4	1	-	-	-	-	4

Our final example will look into the case described in the procedure where none of the 3 cases in the left looking case work. We are trying to remove the conflict in the seventh column block.

①	2	3	4	1	②	3	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	2	③	4	4
-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	-	-	-	-	-	-	-	-	2	1	4	③	4
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	1	2	③	4	4
3	2	①	4	1	3	②	4	-	-	-	-	-	-	-	-	-	-	-	-	1	2	③	4	-	-	-	-	4
3	2	4	①	-	-	-	-	1	3	②	4	-	-	-	-	-	-	-	-	1	2	4	③	-	-	-	-	4
-	-	-	-	-	-	-	-	2	3	4	①	1	3	②	4	2	1	4	③	-	-	-	-	-	-	-	-	4
-	-	-	-	3	2	4	①	-	-	-	-	1	2	4	③	1	2	③	4	-	-	-	-	-	-	-	-	4

Here we are in Case 3, and we can't do any block swapping to improve the situtation directly, so we choose the first block row, and the first block column, we swap the 1 and the 4 in the fourth and fifth rows, and then look to the next block along in these rows.

①	2	3	4	1	②	3	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	2	③	4	4
-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	-	-	-	-	-	-	-	-	2	1	4	③	4
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	1	2	③	4	4
3	2	1	④	1	3	②	4	-	-	-	-	-	-	-	-	-	-	-	-	1	2	③	4	-	-	-	-	1
3	2	④	1	-	-	-	-	1	3	②	4	-	-	-	-	-	-	-	-	1	2	4	③	-	-	-	-	1
-	-	-	-	-	-	-	-	2	3	4	①	1	3	②	4	2	1	4	③	-	-	-	-	-	-	-	-	4
-	-	-	-	3	2	4	①	-	-	-	-	1	2	4	③	1	2	③	4	-	-	-	-	-	-	-	-	4

We see that in the fourth row of the second block column, we can perform a Spare Colour Swap to swap the 2 for the 1, this then puts the seventh block into case 2, as the legal colour in the first row of the second block is now the same colour as the colour selected in the first row of the seventh block, so we can perform the steps in case 2 and remove the conflict in the seventh block.

①	2	3	4	1	2	③	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	②	3	4	4
-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	-	-	-	-	-	-	-	-	2	1	4	③	4
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	①	2	3	4	1	②	3	4	1	2	③	4	4
3	2	1	④	①	3	2	4	-	-	-	-	-	-	-	-	-	-	-	-	1	2	③	4	-	-	-	-	1
3	2	④	1	-	-	-	-	1	3	②	4	-	-	-	-	-	-	-	-	1	2	4	③	-	-	-	-	1
-	-	-	-	-	-	-	-	2	3	4	①	1	3	②	4	2	1	4	③	-	-	-	-	-	-	-	-	4
-	-	-	-	3	2	4	①	-	-	-	-	1	2	4	③	1	2	③	4	-	-	-	-	-	-	-	-	4

Hopefully with these examples you now have a good idea of how the procedure works, and give you an idea of why the final case which I was unable to justify should always be able to work. Some statements within the procedure are unique to the maximum vertex degree 3 case, however it's not too difficult to imagine they will have an analogue for higher degree cases. I will now look into what is different in the maximum vertex degree 4 case.

4.3.2 In the Maximum Vertex Degree 4 Case

When we have maximum vertex degree 4, we can still do spare colour swaps as earlier, if they are possible, however we no longer have that spare colour swaps being impossible implies that all the chosen colours and spare colours lie within two columns of the block we are sorting. Instead we know that spare colours swaps being impossible means that

the spare and chosen colours lie within 3 columns of the block we're sorting, this results in another issue, in the maximum vertex degree 3 case, only the spare colours and chosen colours lie within the 2 columns, whereas in this case, there will be some colours in the 3 columns that aren't spare or chosen colours. The following procedure works similar to the maximum vertex 3 case but taking into account the 3 column issue:

Input Shortened 4-Block 5-Distortion Matrix, M, representing a connected bipartite graph (we can add this connected condition without loss of generality, if the graph is not connected, we can consider them as two graphs and run the procedure individually on each component)

Output Block Traversal of M (except in certain cases discussed below)

As in the first procedure, we begin by going through each block row and selecting 1 from the first non-empty block (again will refer to these as blocks from here, and refer to empty blocks explicitly) and then 2 from the next, 3 from the one after that and 4 from the fourth, we also add the spare colour column to the end of the row, and assign it's value as 5.
We then begin to go through each block column, β, of M from left to right, removing all conflicts using the following methods (attempted in this order):

Spare Colour Swap Exactly the same as last time, for each block within β try swapping some of the current selection with their respective rows spare colour, then update the spare colour.

NOTE As mentioned above, this fails if the spare colours and selected colours lie entirely within 3 columns of β.

Right Block Swap Again, this is exactly the same as last time, we swap our selected elements in β and change the selected elements in blocks to the right of β to remove all conflicts.

NOTE As last time, we obviously can't do this if there are no blocks to the right of β involving at least one of the block rows of β. We move onto the cases where we look left now.

Cases When We Look Left There are significantly more cases this time:

1. If all the spare colours and selected colours lie within 2 columns of β then we are in the situation where the 2 columns can contain only the spare and selected colours, as such, we can follow the maximum vertex degree 3 case and we can make it so that we swap at least one selected colour out of these two columns to enter the 3 column case.

2. In the column case, we have the spare colours, the selected colours, and 4 other colours, we shall call these colours the 'unselectable colours'. Each block, B, of

β will of course be in a block row with 3 blocks to the left of it, we shall refer to these blocks as B_1, B_2 and B_3. We now test each of the following on each B in β which are involved in conflicts:

If (legal colour in B_i for some $i \in \{1,2,3\}$ = selected colour in B) Try

If (Unselectable colour in B $\neq$ Selected colour in B_i) Try

This means that the we can swap the Selected colour in B_i for the legal colour in B_i and the selected colour in B for one of the legal colours in B, and then we are done.

If (Unselectable colour in B = Selected colour in B_i) Try

If (selected in B_i = legal in B_k, $k \in \{1,2,3\} \setminus i$) Try

Then selected colour in B_i equals legal colour in B, and we can swap selected in B_i for legal in B_i, selected in B_k for legal in B_k and selected in B for legal in B, and we're done.

If (selected in $B_i \neq$ legal in B_k and B_j, $j,k \in \{1,2,3\} \setminus i$) Try

Then the selected in B_j = legal in B_k, and selected in B_k = legal in B_j. This is unfortunately back to the case in maximum vertex degree 3, where I was unable to provide further processes within the procedure to remove the conflict. The idea remains that we can examine a route which involves several block rows connecting block columns together leading back to β, where we can do spare colours swaps in order to remove the conflict, and failing that, we can go back, swap the two colours described in B_j and B_k and this will then allow us to bypass the previous failures, but as said, I could not prove that this step would succeed.

If (legal colour in B_i for some $i \in \{1,2,3\} \neq$ selected colour in B) Try

This case, in this case, as above it is unproven that I can complete the colouring process here, the idea is the same as mentioned above, except in this case we will be able to 'cycle' between the selected and free colours of B_i, B_j and B_k.

This completes the procedure. The problem we are up against at the end of this algorithm is exactly the same as the maximum vertex degree 3 case, and so a solution to that problem would provide a proof of both 4-proper distortion colouring of simple bipartite graphs of maximum vertex degree 3 (as mentioned this has already been proved in [1]) and 5-proper distortion colouring of simple bipartite graphs of maximum vertex degree 4. Similarly, we can see that the problem extended to the maximum vertex degree 4 cases relatively simply, and again it's very reasonable to believe such a procedure would work on maximum vertex

degree 5 and higher cases, albeit with the same problem that this procedure comes up against in certain cases.

5 Conclusion

This project gave a strong insight into the complications that arise in distortion colourings, even in the case of simple graphs. We attempted to create a procedure that would allow us to provide distortion colourings, however we saw that due to the fact we must look further than the directly adjacent vertices when we get to the end, which of course makes sense, but I was unsure of how to include that within the procedure.

Bibliography

[1] Agelos

[2] Noga

[3] Graph Theory

[4] http://users.dcc.uchile.cl/ raparede/clases/iet121/cormen.pdf

[5] http://0-search.proquest.com.pugwash.lib.warwick.ac.uk/docview/918506217/191BE1EB5C9E4D5