Rainbow Colouring of Split and Threshold Graphs

L. Sunil Chandran and Deepak Rajendraprasad

Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore -560012, India.
{*sunil, deepakr*}*@csa.iisc.ernet.in*

May 9, 2012

Abstract

A *rainbow colouring* of a connected graph is a colouring of the edges of the graph, such that every pair of vertices is connected by at least one path in which no two edges are coloured the same. Such a colouring using minimum possible number of colours is called an *optimal rainbow colouring*, and the minimum number of colours required is called the *rainbow connection number* of the graph. A *Chordal Graph* is a graph in which every cycle of length more than 3 has a chord. A *Split Graph* is a chordal graph whose vertices can be partitioned into a clique and an independent set. A *threshold graph* is a split graph in which the neighbourhoods of the independent set vertices form a linear order under set inclusion. In this article, we show the following:

1. The problem of deciding whether a graph can be rainbow coloured using 3 colours remains NP-complete even when restricted to the class of split graphs. However, any split graph can be rainbow coloured in linear time using at most one more colour than the optimum.
2. For every integer $k \geq 3$, the problem of deciding whether a graph can be rainbow coloured using k colours remains NP-complete even when restricted to the class of chordal graphs.
3. For every positive integer k, threshold graphs with rainbow connection number k can be characterised based on their degree sequence alone. Further, we can optimally rainbow colour a threshold graph in linear time.

Keywords: rainbow connectivity, rainbow colouring, threshold graphs, split graphs, chordal graphs, degree sequence, approximation, complexity.

1 Introduction

Connectivity is one of the basic concepts of graph theory. It plays a fundamental role both in theoretical studies and in applications. When a network (transport, communication, social, etc) is modelled as a graph, connectivity gives a way of quantifying its robustness. This may be the reason why connectivity is possibly the problem that has been studied on the largest variety of computational models [25]. Due to the diverse application requirements and manifold theoretical interests, many variants of the connectivity problem have

been studied. One typical case is when there are different possible types of connections (edges) between nodes and additional restrictions on connectivity based on the types of edges that can used in a path. In this case we can model the network as an edge-coloured graph. One natural restriction to impose on connectivity is that any two nodes should be connected by a path in which no edge of the same type (colour) occurs more than once. This is precisely the property called *rainbow connectivity*. Such a restriction for the paths can arise, for instance, in routing packets in a cellular network with transceivers that can operate in multiple frequency bands or in routing secret messages between security agencies using different handshaking passwords in different links [18] [5]. The problem was formalised in graph theoretic terms by Chartrand et al. [7] in 2008.

An *edge colouring* of a graph is a function from its edge set to the set of natural numbers. A path in an edge coloured graph with no two edges sharing the same colour is called a *rainbow path*. An edge coloured graph is said to be *rainbow connected* if every pair of vertices is connected by at least one rainbow path. Such a colouring is called a *rainbow colouring* of the graph. A rainbow colouring using minimum possible number of colours is called *optimal*. The minimum number of colours required to rainbow colour a connected graph is called its *rainbow connection number*, denoted by $rc(G)$. For example, the rainbow connection number of a complete graph is 1, that of a path is its length, that of an even cycle is its diameter, that of an odd cycle of length at least 5 is one more than its diameter, and that of a tree is its number of edges. Note that disconnected graphs cannot be rainbow coloured and hence the rainbow connection number for them is left undefined. Any connected graph can be rainbow coloured by giving distinct colours to the edges of a spanning tree of the graph. Hence the rainbow connection number of any connected graph is less than its number of vertices.

While formalising the concept of rainbow colouring, Chartrand et al. also determined the precise values of rainbow connection number for some special graphs [7]. Subsequently, there have been various investigations towards finding good upper bounds for rainbow connection number in terms of other graph parameters [4] [21] [15] [24] [2] and for many special graph classes [19] [24] [2] [3]. Behaviour of rainbow connection number in random graphs is also well studied [4] [11] [23] [9]. A basic introduction to the topic can be found in Chapter 11 of the book *Chromatic Graph Theory* by Chartrand and Zhang [6] and a survey of most of the recent results in the area can be found in the article by Li and Sun [18] and also in their forthcoming book *Rainbow Connection of Graphs* [17].

On the computational side, the problem has received relatively less attention. It was shown by Chakraborty et al. that computing the rainbow connection number of an arbitrary graph is NP-Hard [5]. In particular, it was shown that the problem of deciding whether a graph can be rainbow coloured using 2 colours is NP-complete. Later, Ananth et al. [1] complemented the result of Chakraborty et al., and now we know that for every integer $k \geq 2$, it is NP-complete to decide whether a given graph can be rainbow coloured using k colours. Chakraborty et al., in the same article, also showed that deciding whether a given edge coloured graph is rainbow connected is NP-complete. It was then shown by Li and Li that this problem remains NP-complete even when restricted to the class of bipartite graphs [16].

On the positive side, Basavaraju et al. have demonstrated an $O(nm)$-time $(r+3)$-factor approximation algorithm for rainbow colouring any graph with radius r [2]. Constant factor approximation algorithms for rainbow colouring Cartesian, strong and lexicographic products of non-trivial graphs are reported in [3]. Constant factor approximation algorithms for bridgeless chordal graphs, and additive approximation algorithms for interval, AT-free, threshold and circular arc graphs without pendant vertices will follow from the proofs of their upper bounds [24]. To the best of our knowledge, no efficient optimal

rainbow colouring algorithm has been reported for any non-trivial subclass of graphs.

1.1 Our Results

In this article we consider the problem of rainbow colouring split graphs and a particular subclass of split graphs called threshold graphs (Definition 3). We show the following results.

1. The problem of deciding whether a graph can be rainbow coloured using 3 colours remains NP-complete even when restricted to the class of split graphs (Corollary 5). Any split graph can be rainbow coloured in linear time using at most one more colour than the optimum (Algorithm 1).

This is similar to the problem of finding the chromatic index of a graph. Though every graph with maximum degree Δ can be properly edge-coloured in $O(nm)$ time using $\Delta+1$ colours using a constructive proof of Vizing's Theorem [20], it is NP-hard to decide whether the graph can be coloured using Δ colours [12].

No two pendant edges (Definition 2) can share the same colour in any rainbow colouring of a graph (Observation 2). The +1-approximation algorithm above is obtained by carefully reusing the same colours on most of the remaining edges of the graph. The hardness result is obtained by demonstrating a reduction from the problem of 3-colourability of 3-uniform hypergraphs. In fact, the technique in the reduction can be extended to show the following result for chordal graphs.

2. For every integer $k \geq 3$, the problem of deciding whether a graph can be rainbow coloured using k colours remains NP-complete even when restricted to the class of chordal graphs (Theorem 6).

Though a similar hardness result is known for deciding the rainbow connection number of general graphs, the above strengthening to chordal graphs is interesting since, unlike for general graphs, a constant factor approximation algorithm is already known for rainbow colouring chordal graphs. Chandran et al. [24] have shown that any bridgeless chordal graph can be rainbow coloured using at most $3r$ colours, where r is the radius of the graph. The proof given there is constructive and can be easily extended to a polynomial-time algorithm which will colour any chordal graph G with b bridges and radius r using at most $3r+b$ colours. Since $\max\{r, b\}$ is easily seen to be a lower bound for $rc(G)$, this immediately gives us a 4-factor approximation algorithm.

3. For every positive integer k, threshold graphs with rainbow connection number exactly k can be characterised based on their degree sequence (Definition 2) alone (Corollary 14). Further, we can optimally rainbow colour a threshold graph in linear time (Algorithm 4).

In particular we show that if $d_1 \geq \cdots \geq d_n$ is the degree sequence of an n-vertex threshold graph G, then

$$rc(G) = \begin{cases} 1, & d_n = n-1 \\ 2, & d_n < n-1 \text{ and } \sum_{i=k}^{n} 2^{-d_i} \leq 1 \\ \max\{3, p\}, & \text{otherwise} \end{cases} \tag{1}$$

where $k = \min\{i : 1 \leq i \leq n,\, d_i \leq i-1\}$ and $p = |\{i : 1 \leq i \leq n,\, d_i = 1\}|$.

Both the characterisation and the algorithm are obtained by connecting the problem of rainbow colouring a threshold graph to that of generating a prefix-free binary code.

1.2 Preliminaries

All graphs considered in this article are finite, simple and undirected. For a graph G, we use $V(G)$ and $E(G)$ to denote its vertex set and edge set respectively. Unless mentioned otherwise, n and m will respectively denote the number of vertices and edges of the graph in consideration. The shorthand $[n]$ denotes the set $\{1, \ldots, n\}$. The cardinality of a set S is denoted by $|S|$.

Definition 1. Let G be a connected graph. The *length* of a path is its number of edges. The *distance* between two vertices u and v in G, denoted by $d(u, v)$ is the length of a shortest path between them in G. The *eccentricity* of a vertex v is $ecc(v) := \max_{x \in V(G)} d(v, x)$. The *diameter* of G is $diam(G) := \max_{x \in V(G)} ecc(x)$ and *radius* of G is $radius(G) := \min_{x \in V(G)} ecc(x)$.

Definition 2. The *neighbourhood* $N(v)$ of a vertex v is the set of vertices adjacent to v but not including v. The *degree* of a vertex v is $d_v := |N(v)|$. The *degree sequence* of a graph is the non-increasing sequence of its vertex degrees. A vertex is called *pendant* if its degree is 1. An edge incident on a pendant vertex is called a *pendant edge*.

Definition 3. A graph G is called *chordal*, if there is no induced cycle of length greater than 3. A graph G is a *split graph*, if $V(G)$ can be partitioned into a clique and an independent set. A graph G is a *threshold graph*, if there exists a weight function $w : V(G) \to \mathbb{R}$ and a real constant t such that two vertices $u, v \in V(G)$ are adjacent if and only if $w(u) + w(v) \geq t$.

Before getting into the main results, we note two elementary and well known observations on rainbow colouring whose proofs we omit.

Observation 1. *For every connected graph G, we have $rc(G) \geq diam(G)$.*

Observation 2. *If u and v are two pendant vertices in a connected graph G, then their incident edges get different colours in any rainbow colouring of G. In particular, if G has p pendant vertices, then $rc(G) \geq p$.*

2 Split Graphs: Hardness and Approximation Algorithm

We first show that determining the rainbow connection number of a split graph is NP-hard, by demonstrating a reduction to it from the 3-colouring problem on 3-uniform hypergraphs.

Definition 4. A *hypergraph* H is a tuple (V, E), where V is a finite set and $E \subseteq 2^V$. Elements of V and E are called vertices and (hyper-)edges respectively. The hypergraph H is called *r-uniform* if $|e| = r$ for every $e \in E$. An r-uniform hypergraph is called *complete* if $E = \{e \subset V : |e| = r\}$.

Definition 5. Given a hypergraph $H(V, E)$ and a colouring $C_H : V \to \mathbb{N}$, an edge is called *k-coloured* if the edge contains vertices of k different colours. An edge is called *monochromatic* if it is 1-coloured. The colouring C_H is called *proper* if no edge in E is monochromatic under C_H. The minimum number of colours required to properly colour H is called its *chromatic number* and is denoted by $\chi(H)$.

We need a 3-uniform hypergraph of chromatic number 3 to avoid the occurrence of a border case in the reduction. The following observation gives us one.

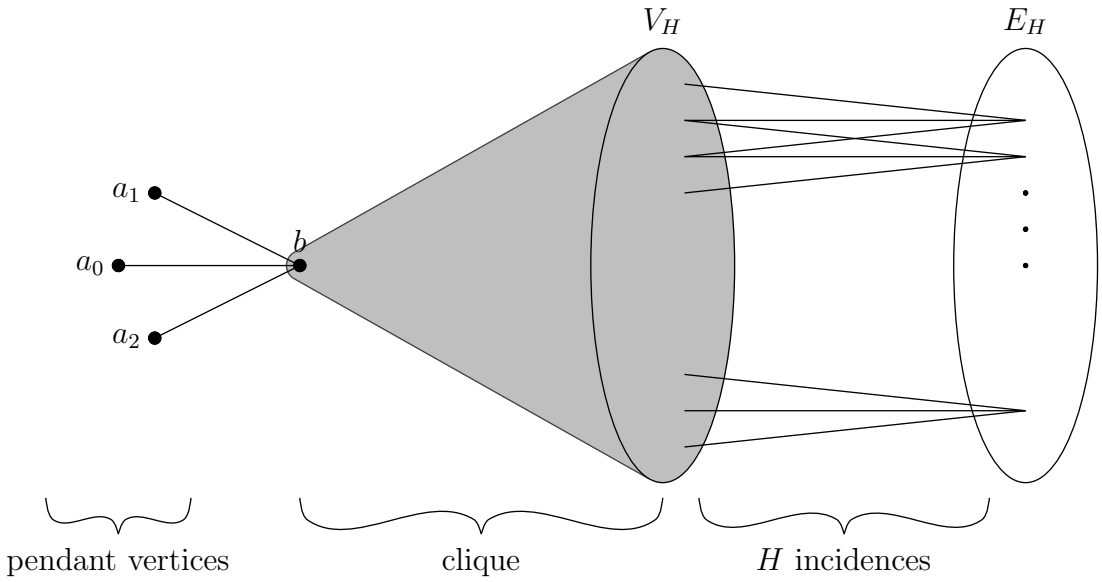

Figure 1: Split graph G constructed from a 3-uniform hypergraph H. Note that $V_H \cup \{b\}$ is a clique and $E_H \cup \{a_0, a_1, a_2\}$ is an independent set in G.

Observation 3. *Let K_5^3 be the complete 3-uniform hypergraph on 5 vertices. Then $\chi(K_5^3) = 3$.*

Proof. Assign colours $0, 0, 1, 1, 2$ to the 5 vertices of K_5^3. This is a proper colouring of K_5^3 since every edge contains 3 vertices and hence cannot be monochromatic. On the other hand, in any colouring of K_5^3 using fewer than 3 colours, some three vertices have to share the same colour and hence the edge of K_5^3 constituted of those 3 vertices will be monochromatic. Hence $\chi(K_5^3) = 3$. □

It follows from Theorem 1.1 in [13] that it is NP-hard to decide whether an n-vertex 3-uniform hypergraph can be properly coloured using 3 colours. A reduction from this problem to a problem of computing the rainbow connection number of a split graph is illustrated in the proofs of Theorem 4 and Theorem 6.

Theorem 4. *The first problem below* (P1) *is polynomial-time reducible to the second* (P2).

P1. *Given a 3-uniform hypergraph H', decide whether $\chi(H') \leq 3$.*

P2. *Given a split graph G, decide whether $rc(G) \leq 3$.*

Proof. Let H be the disjoint union of H' and a complete 3-uniform hypergraph on 5 vertices (K_5^3). This ensures that $\chi(H) \geq 3$ (Observation 3) and that $\chi(H) = 3$ iff $\chi(H') \leq 3$. Let V_H and E_H be the vertex set and edge set, respectively, of H. We construct a graph $G(V_G, E_G)$ from $H(V_H, E_H)$ as follows (See Figure 1).

$$V_G = V_H \cup E_H \cup \{a_0, a_1, a_2, b\} \tag{2}$$

$$\begin{aligned} E_G = {} & \{\{v, e\} : v \in V_H, e \in E_H, v \in e \text{ in } H\} \\ & \cup \{\{v, v'\} : v, v' \in V_H, v \neq v'\} \\ & \cup \{\{b, v\} : v \in V_H\} \\ & \cup \{\{a_i, b\} : i = 0, 1, 2\} \end{aligned} \tag{3}$$

The graph G thus constructed is a split graph with $V_H \cup \{b\}$ being a clique and its complement with respect to V_G, which is $E_H \cup \{a_0, a_1, a_2\}$, being an independent set. It is clear that G can be constructed from H' in polynomial-time. We complete the proof by showing that $\chi(H) = 3$ iff $rc(G) = 3$.

Firstly, we show that if $rc(G) = 3$, then $\chi(H) = 3$. Since $\chi(H) \geq 3$, it suffices to show that H can be properly 3-coloured. Let $C_G : E_G \to \mathbb{Z}_3$ be a rainbow colouring of G. Define a colouring $C_H : V_H \to \mathbb{Z}_3$ by $C_H(v) = C_G(\{b, v\})$ for each $v \in V_H$. We claim that C_H is a proper colouring of H. For the sake of contradiction, suppose that one of the hyper-edges e_H of H is monochromatic under C_H, i.e, all the vertices in e_H get the same colour j for some $j \in \mathbb{Z}_3$. This happens only when $C_G(\{b, v\}) = j$, $\forall v \in e_H$. Hence all the paths of length two from b to e_H in G will use the colour j. Since $\{a_0, a_1, a_2\}$ are pendant vertices, the edges from $\{a_0, a_1, a_2\}$ to b all have distinct colours in any rainbow colouring of G (Observation 2). Hence one of them, say $\{a_i, b\}$, gets the colour j. Then it is easy to see that there is no rainbow path from a_i to e_H in G under C_G (Note that any rainbow path in a 3-coloured graph has length at most 3). This contradicts the fact that C_G was a rainbow colouring of G.

Next, we show that if $\chi(H) = 3$, then $rc(G) = 3$. Since G has 3 pendant vertices, $rc(G) \geq 3$ (Observation 2). So it suffices to show that G can be rainbow coloured using 3 colours. Let $C_H : V_H \to \mathbb{Z}_3$ be a proper colouring of H. Let $V_i = \{v \in V_H : C_H(v) = i\}$, $i \in \mathbb{Z}_3$, be the colour classes. Note that none of the colour classes is empty as $\chi(H) = 3$. We define a colouring $C_G : E_G \to \mathbb{Z}_3$ as follows (See Figure 2). $C_G(\{b, v\}) = C_H(v)$ for each $v \in V_H$. Consider a hyper-edge $e_H = \{v_0, v_1, v_2\}$ of H. If e_H is 3-coloured in C_H then $C_G(\{v_i, e_H\}) = C_H(v_i) + 1$ (Note that the colours are from $\mathbb{Z}_3$ and hence the addition is modulo 3). If e_H is 2-coloured in H, then without loss of generality, let $C_H(v_0) = C_H(v_1) = i$ and $C_H(v_2) = j$, $j \neq i$. Set $C_G(\{v_0, e_H\}) = i + 1$, $C_G(\{v_1, e_H\}) = i + 2$, and $C_G(\{v_2, e_H\}) \in \mathbb{Z}_3 \setminus \{i, j\}$. This ensures that for every hyper-edge $e \in E_H$, for each colour $i \in \mathbb{Z}_3$, there exists a 2-length rainbow path $P_{e,i}$ from b to e such that colour i does not appear in path $P_{e,i}$. The remaining edges of G are coloured as follows.

$$\begin{aligned}
C_G(\{a_i, b\}) &= i \quad \forall i \in \mathbb{Z}_3 \\
C_G(\{v, v'\}) &= i \quad \forall v, v' \in V_i,\ v \neq v',\ \forall i \in \mathbb{Z}_3 \\
C_G(\{v, v'\}) &= 2 \quad \forall v \in V_0, v' \in V_1 \cup V_2 \\
C_G(\{v, v'\}) &= 0 \quad \forall v \in V_1, v' \in V_2.
\end{aligned} \tag{4}$$

We show that C_G is a rainbow colouring of G by demonstrating a rainbow path between every pair of non adjacent vertices in G. First we demonstrate the paths from $\{a_0, a_1, a_2, b\}$ to all their non-adjacent vertices. (The numbers above an edge indicate the colour assigned to the edge under C_G.)

$$\begin{aligned}
a_i \text{ to } a_j, i \neq j &: a_i \overset{i}{\text{---}} b \overset{j}{\text{---}} a_j \\
a_i \text{ to } v_j \in V_j, i \neq j &: a_i \overset{i}{\text{---}} b \overset{j}{\text{---}} v_j \\
a_i \text{ to } v_i \in V_i &: a_0 \overset{0}{\text{---}} b \overset{1}{\text{---}} V_1 \overset{2}{\text{---}} v_0 \\
&\quad\ a_1 \overset{1}{\text{---}} b \overset{0}{\text{---}} V_0 \overset{2}{\text{---}} v_1 \\
&\quad\ a_2 \overset{2}{\text{---}} b \overset{1}{\text{---}} V_1 \overset{0}{\text{---}} v_2 \\
a_i \text{ to } e \in E_H &: a_i \overset{i}{\text{---}} b \overset{P_{e,i}}{\text{---}} e \\
b \text{ to } e \in E_H &: b \overset{P_{e,0}}{\text{---}} e
\end{aligned} \tag{5}$$

The rainbow path between any vertex $v \in V_H$ and a non-adjacent vertex $e \in E_H$ is given by $v \overset{i}{\text{---}} b \overset{P_{e,i}}{\text{---}} e$, if $v \in V_i$. It remains to demonstrate a rainbow path between

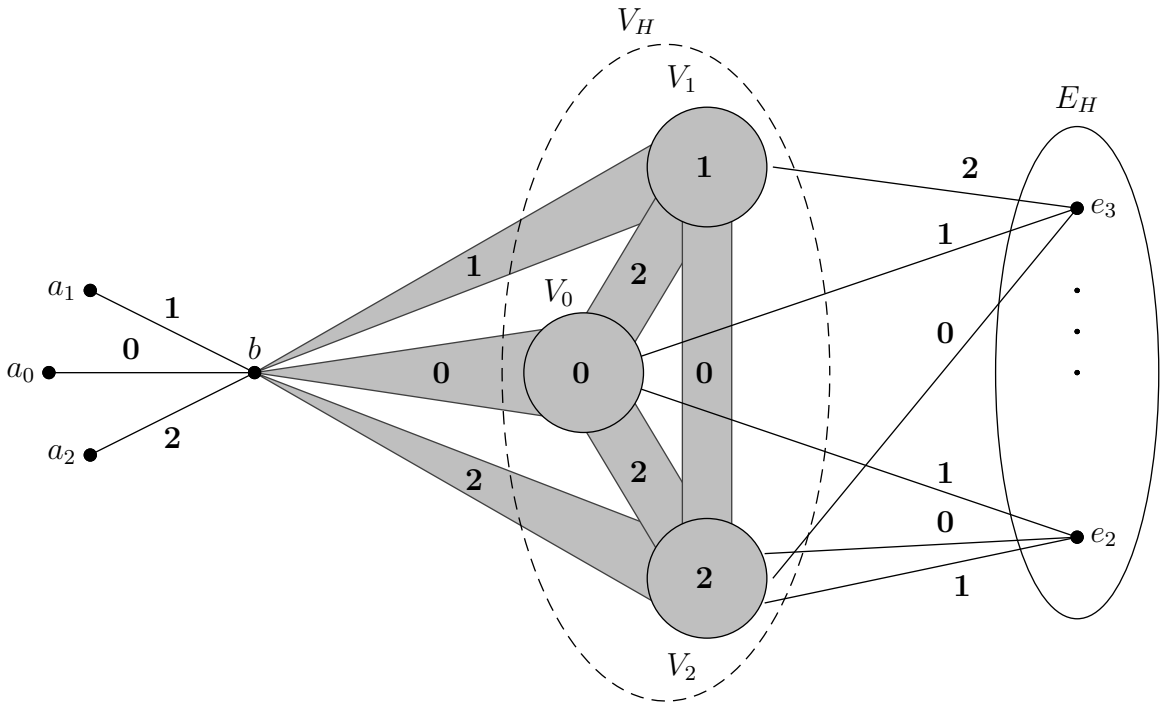

Figure 2: Rainbow colouring of split graph G based on the 3-colouring of hypergraph H. In the figure, e_3 is a sample 3-coloured edge and e_2 is a sample 2-coloured edge.

any two vertices $e = \{v_0, v_1, v_2\}, e' = \{v'_0, v'_1, v'_2\} \in E_H$. By $C_H(e)$ we denote the 3-tuple $(C_H(v_0), C_H(v_1), C_H(v_2))$. If e is 3-coloured, we relabel $\{v_0, v_1, v_2\}$ so that $C_H(e) = (0, 1, 2)$ and hence $C_G(\{v_i, e\}) = i + 1$. If e is 2-coloured, we relabel $\{v_0, v_1, v_2\}$ so that $C_H(e) = (i, i, j), j \neq i$ and such that $C_G(\{v_0, e\}) = i + 1$ and $C_G(\{v_1, e\}) = i + 2$. We do the same for e' too. Edges e and e' may share some vertices, in which case the same vertex will get different labels when considered under e and e'. We consider the following cases separately: (i) both e and e' are 3-coloured, (ii) e is 3-coloured and e' is 2-coloured and (iii) both e and e' are 2-coloured. The last case is further split into 4 sub-cases.

$$
\begin{aligned}
C_H(e) = C_H(e') = (0,1,2) &: e \overset{1}{\text{---}} v_0 \overset{2}{\text{---}} v'_2 \overset{0}{\text{---}} e' \quad \text{(Case i)} \\
C_H(e) = (0,1,2), C_H(e') = (i,i,j), i \neq j &: e \overset{i+1}{\text{---}} v_i \overset{i}{\text{---}} v'_1 \overset{i+2}{\text{---}} e' \quad \text{(Case ii)} \\
C_H(e) = (i,i,j), C_H(e') = (i,i,k) &: e \overset{i+1}{\text{---}} v_0 \overset{i}{\text{---}} v'_1 \overset{i+2}{\text{---}} e' \quad \text{(Case iii)} \\
C_H(e) = (0,0,j), C_H(e') = (1,1,k) &: e \overset{1}{\text{---}} v_0 \overset{2}{\text{---}} v'_1 \overset{0}{\text{---}} e' \\
C_H(e) = (1,1,j), C_H(e') = (2,2,k) &: e \overset{2}{\text{---}} v_0 \overset{0}{\text{---}} v'_1 \overset{1}{\text{---}} e' \\
C_H(e) = (2,2,j), C_H(e') = (0,0,k) &: e \overset{0}{\text{---}} v_0 \overset{2}{\text{---}} v'_0 \overset{1}{\text{---}} e' \qquad (6)
\end{aligned}
$$

It is possible that v_i may coincide with v'_1 in Case (ii), and v_0 may coincide with v'_1 in the first sub-case of Case (iii). In both those situations, we still get a 2-length rainbow path between the end points without using the middle edge indicated above. We have exhausted all the cases and hence C_G is a rainbow colouring of G. □

Since Problem P1 is known to be NP-hard, so is Problem P2. Further, it is easy to see that the problem P2 is in NP. Hence the following corollary.

Corollary 5. *Deciding whether $rc(G) \leq 3$ remains NP-complete even when G is restricted to be in the class of split graphs.*

The reduction used in the proof of Theorem 4 can be extended to show that for every $k \geq 3$, it is NP-complete to decide whether a chordal graph can be rainbow coloured using k colours.

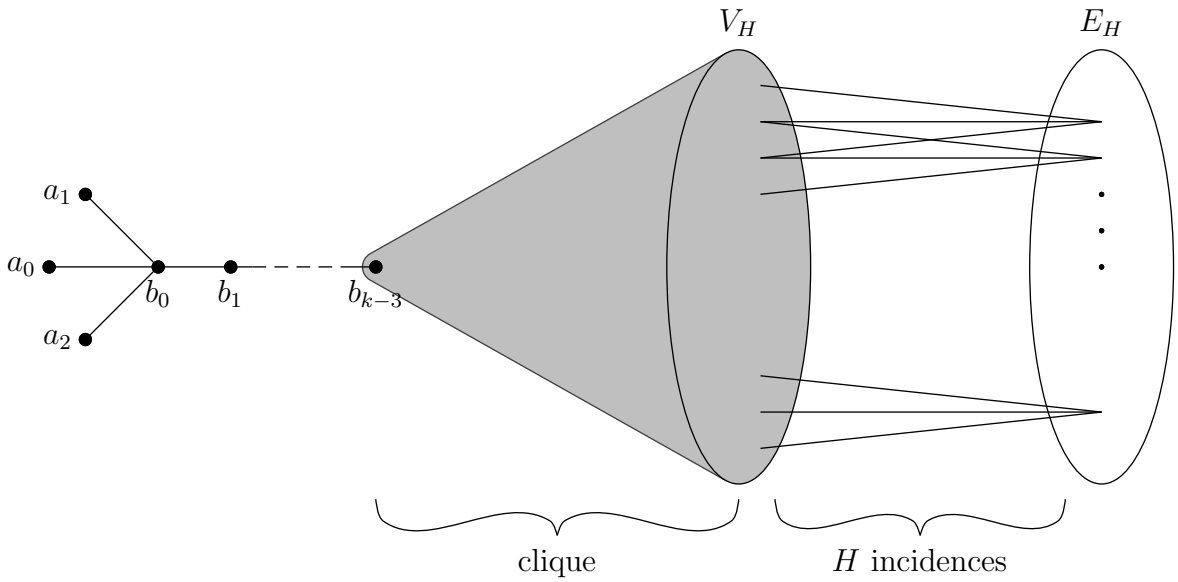

Figure 3: Chordal graph G_k of diameter k constructed from a 3-uniform hypergraph H.

Theorem 6. *For any integer $k \geq 3$, the first problem below* (P1) *is polynomial-time reducible to the second* (P2)*.*

P1. *Given a 3-uniform hypergraph H', decide whether $\chi(H') \leq 3$.*

P2. *Given a chordal graph G, decide whether $rc(G) \leq k$.*

In particular, for every integer $k \geq 3$, the problem of deciding whether $rc(G) \leq k$ remains NP-complete even when G is restricted to be in the class of chordal graphs.

Proof. Let H be the disjoint union of H' and a complete 3-uniform hypergraph on 5 vertices (K_5^3). This ensures that $\chi(H) \geq 3$ (Observation 3) and that $\chi(H) = 3$ iff $\chi(H') \leq 3$. Let V_H and E_H be the vertex set and edge set, respectively, of H. Let $k \geq 3$ be fixed. We construct a graph $G_k(V_G, E_G)$ from H as follows (See Figure 3).

$$
\begin{aligned}
V_G &= V_H \cup E_H \cup \{a_0, a_1, a_2, b_0, \ldots, b_{k-3}\} && (7)\\
E_G &= \{\{v, e\} : v \in V_H, e \in E_H, v \in e \text{ in } H\} \\
&\quad \cup \{\{v, v'\} : v, v' \in V_H, v \neq v'\} \\
&\quad \cup \{\{b_{k-3}, v\} : v \in V_H\} \\
&\quad \cup \{\{b_{i-1}, b_i\} : i = 1, \ldots, k-3\} \\
&\quad \cup \{\{a_i, b_0\} : i = 0, 1, 2\} && (8)
\end{aligned}
$$

The graph G thus constructed is easily seen to be a chordal graph with diameter k. It is clear that G can be constructed from H' in polynomial-time. We complete the proof by showing that $\chi(H) = 3$ iff $rc(G) = k$.

It is easy to see that when $k = 3$, the graph G_3 constructed as above is the same as the split graph constructed in the proof of Theorem 4. In that proof we showed a rainbow colouring of G_3 using 3 colours in the case when $\chi(H) = 3$. The same colouring can be extended to G_k by giving $k-3$ new colours exclusively to the edges $\{b_{i-1}, b_i\}$, $i = 1, \ldots, k-3$. Since the original 3-colouring made G_3 rainbow connected, it is easy to see that this colouring makes G_k rainbow connected. Hence it is enough to show that if $rc(G_k) = k$, then $\chi(H) = 3$.

Since $\chi(H) \geq 3$, it suffices to show that H can be properly 3-coloured. Let $C_G : E_G \to \{0, \ldots, k-1\}$ be a rainbow colouring of G_k. Since the subgraph T_k of G_k induced

on $\{a_0, a_1, a_2, b_0, \ldots, b_{k-3}\}$ is a tree with k edges, it is easy to see that in any rainbow colouring of G_k the edges of T_k get k distinct colours. Without loss of generality we rename the colours so that $C_G(\{a_i, b_0\}) = i$, $i \in \{0, 1, 2\}$. Hence the edges in the path from b_0 to b_{k-3} get colours from $\{3, \ldots, k-1\}$. Define a colouring $C_H : V_H \to \{0, 1, 2\}$ by $C_H(v) = \min\{C_G(\{b_{k-3}, v\}), 2\}$ for each $v \in V_H$. We claim that C_H is a proper colouring of H. For the sake of contradiction, suppose that one of the hyper-edges e_H of H is monochromatic under C_H, i.e, all the vertices in e_H get the same colour j for some $j \in \{0, 1, 2\}$. This happens only when $\min\{C_G(\{b_{k-3}, v\}), 2\} = j$, $\forall v \in e_H$. If j is 0 or 1, all the paths of length two from b_{k-3} to e_H in G will use the colour j and hence there is no (k-length) rainbow path from a_j to e_H. If j is 2, then too, all the paths of length two from b_{k-3} to e_H in G will use one of the colours already used in the unique path from a_2 to b_{k-3} and hence there is no (k-length) rainbow path from a_2 to e_H.

Since Problem P1 is known to be NP-hard, so is Problem P2. Further, it is easy to see that the problem P2 is in NP. Hence the result. □

In the wake of Corollary 5, it is unlikely that there exists a polynomial-time algorithm to optimally rainbow colour split graphs in general. In Section 3, we show that the problem is efficiently solvable when restricted to threshold graphs, which are a subclass of split graphs. Before that, we describe a linear-time (approximation) algorithm which rainbow colours any split graph using at most one colour more than the optimum (Theorem 7). First we note that it is easy to find a maximum clique in a split graph, as follows.

The vertices of a graph can be sorted according to their degrees in $O(n)$ time using a counting sort [22]. If $G([n], E)$ is a split graph with the vertices labelled so that $d_1 \geq \cdots \geq d_n$, where d_i is degree of vertex i, then $\{i \in V(G) : d_i \geq i - 1\}$ is a maximum clique in G and $\{i \in V(G) : d_i \leq i - 1\}$ is a maximum independent set in G [10]. Hence we can assume, if needed, that a maximum clique or a maximum independent set or an ordering of the vertices according to their degrees is given as input to our algorithms.

Algorithm 1 ColourSplitGraph

Input: $G([n], E)$, a connected split graph with a maximum clique C.

Output: A rainbow colouring $C_G : E(G) \to \{0, \ldots, \max\{p, 2\}\}$, where p is the number of pendant vertices in $V(G) \setminus C$.

1: $I \leftarrow V(G) \setminus C$ // I is an independent set in G
2: $P \leftarrow \{i \in I : d_i = 1\}$, $p \leftarrow |P|$ // P is the set of pendant vertices in I
3: $C_G(e) \leftarrow 0$, for all edges e with both end points in C.
4: $C_G(e_i) \leftarrow i$ for each pendant edge $e_1, \ldots, e_p$
5: **for** $i \in I \setminus P$ **do**
6: Let $\{e_1, \ldots, e_{d_i}\}$ be the edges incident on i
7: $C_G(e_1) \leftarrow 1$
8: $C_G(e) \leftarrow 2$ for every other edge e incident on i
9: **end for** // Now every vertex in $I \setminus P$ has a 1-coloured and a 2-coloured edge to C
10: **return** C_G

Theorem 7. *For every connected split graph G, Algorithm 1* (ColourSplitGraph) *rainbow colours G using at most $rc(G) + 1$ colours. Further, the time-complexity of Algorithm 1 is $O(m)$.*

Proof. If G is a clique, then $C = V(G)$ and Algorithm 1 colours every edge of G with colour 0. This is an optimal rainbow colouring for G. Hence we can assume that G is not a clique in the following discussions. So $d := diam(G) \geq 2$. It is easy to check, by considering all

pairs of non-adjacent vertices, that Algorithm 1 indeed produces a rainbow colouring of G. For example, between two vertices $v, v' \in I \setminus P$, we get a rainbow path $v \overset{1}{\text{---}} C \overset{0}{\text{---}} C \overset{2}{\text{---}} v'$. It is also evident that the algorithm uses at most $k := \max\{p+1, 3\}$ colours. By Observation 1 and Observation 2, $rc(G) \geq \max\{p, d\} \geq \max\{p, 2\} = k - 1$. Hence the rainbow colouring produced by Algorithm 1 uses at most $rc(G) + 1$ colours.

Further, the algorithm visits each edge exactly once and hence the time-complexity is $O(m)$. □

The following bounds follow directly from Observation 1, Observation 2, and Theorem 7.

Corollary 8. *For every connected split graph G with p pendant vertices and diameter d,*

$$\max\{p, d\} \leq rc(G) \leq \max\{p+1, 3\}.$$

3 Threshold Graphs: Characterisation and Exact Algorithm

Threshold graphs form a subclass of split graphs (Observation 9b). The neighbourhoods of vertices in a maximum independent set of a threshold graph form a linear order under set inclusion (Observation 9c). We exploit this structure to give a full characterisation of rainbow connection number of threshold graphs based on degree sequences (Corollary 14). We use this characterisation to design a linear-time algorithm to optimally rainbow colour any threshold graph (Algorithm 4).

The following observations are easy to make from the definition of a threshold graph (Definition 3).

Observation 9. *Let $G([n], E)$ be a threshold graph with a weight function $w : V(G) \to \mathbb{R}$. Let the vertices be labelled so that $w(1) \geq \cdots \geq w(n)$. Then*

(a) *$d_1 \geq \cdots \geq d_n$, where d_i is the degree of vertex i.*

(b) *$I = \{i \in V(G) : d_i \leq i - 1\}$ is a maximum independent set G and $V(G) \setminus I$ is a clique in G. In particular, every threshold graph is a split graph.*

(c) *$N(i) = \{1, \ldots, d_i\}$, for every $i \in I$. Thus the neighbourhoods of vertices in I form a linear order under set inclusion. Further, if G is connected, then every vertex in G is adjacent to 1.*

Definition 6. A *binary codeword* is a finite string over the alphabet $\{0, 1\}$ (bits). The *length* of a codeword b, denoted by $length(b)$, is the number of bits in the string b. We denote the i-th bit of b by $b(i)$. A codeword b_1 is said to be a *prefix* of a codeword b_2 if $length(b_1) \leq length(b_2)$ and $b_1(i) = b_2(i)$ for all $i \in \{1, \ldots, length(b_1)\}$. A *binary code* is a set of binary codewords. A binary code B is called *prefix-free* if no codeword in B is a prefix of another codeword in B.

The Kraft's Inequality [14] gives a necessary and sufficient condition for the existence of a prefix-free code for a given set of codeword lengths.

Theorem 10 (Kraft 1949 [14])**.** *For every prefix-free binary code $B = \{b_1, \ldots, b_n\}$,*

$$\sum_{i=1}^{n} 2^{-l_i} \leq 1$$

where $l_i = length(b_i)$, and conversely, for any sequence of lengths $l_1, \ldots, l_n$ satisfying the above inequality, there exists a prefix-free binary code $B = \{b_1, \ldots, b_n\}$, with $length(b_i) = l_i$, $i = 1, \ldots, n$.

Observation 11. *Given any sequence of lengths $l_1 \leq \cdots \leq l_n$ satisfying the Kraft Inequality, we can construct a prefix-free binary code $B = \{b_1, \ldots, b_n\}$, with $length(b_i) = l_i$, $i = 1, \ldots, n$ in time $O\big(\sum_{i=1}^{n} l_i\big)$. Further, we can ensure that every bit in b_1 is 0.*

Proof. A *binary tree* is a rooted tree in which every node has at most two child nodes. A node with only one child node is said to be *unsaturated*. The *level* of a node is its distance from the root. We assume that every edge from a parent to its first (second) child, if it exists, is labelled 0 (1). We can represent a prefix-free binary code by a binary tree such that (i) every codeword b_i corresponds to a leaf t_i of the binary tree at level $length(b_i)$ and (ii) the labels on the unique path from the root to a leaf will be the codeword associated with that leaf [8]. We construct a prefix-free binary code with the given length sequence by constructing the corresponding binary tree as explained below.

Create the root, and for every new node created, create its first child till we hit a node t_1 at depth l_1 for the first time. Declare t_1 as a leaf. Once we have created a leaf t_i, $i < n$, we proceed to create the next leaf as follows. Backtrack from t_i along the tree created so far towards the root till we hit the first unsaturated node. Create its second child. If the second child is at level l_{i+1}, then declare it as the leaf t_{i+1}. Else, recursively create first child till we create a node at level l_{i+1} and declare it as leaf t_{i+1}. Terminate this process once we create the leaf t_n.

The process will continue till we create all the n leaves. Otherwise, it has to be the case that every internal node in the tree got saturated by the time we created some leaf t_i, $i < n$. If we have a binary tree T with every internal node saturated, it is easy to see by an inductive argument that $\sum_{t \in L} 2^{-d_t} = 1$, where L is the set of leaves of T and d_t denotes the level of leaf t. Hence $\sum_{j=1}^{n} 2^{-l_j} > \sum_{j=1}^{i} 2^{-l_j} = 1$, contradicting the hypothesis that the lengths $l_1, \ldots, l_n$ satisfy the Kraft Inequality.

It follows from the construction that every bit of b_1 is 0. Since every edge in the tree constructed corresponds to a bit in at least one of the codewords returned, the total number of edges in the tree constructed is at most $\sum_{i=1}^{n} l_i$. Since each edge of the tree is traversed at most twice, the construction will be completed in time $O\big(\sum_{i=1}^{n} l_i\big)$. □

Now we give a necessary and sufficient condition for 2-rainbow-colourability of a threshold graph.

Theorem 12. *For every connected threshold graph $G([n], E)$ with $d_1 \geq \cdots \geq d_n$, $rc(G) \leq 2$ if and only if*

$$\sum_{i=k}^{n} 2^{-d_i} \leq 1, \tag{9}$$

where d_i is the degree of vertex i and $k = \min\{i : 1 \leq i \leq n,\ d_i \leq i - 1\}$. Further, if G satisfies Inequality (9), then Algorithm 2 (ColourThresholdGraph-Case1) gives an optimal rainbow colouring of G in $O(m)$ time.

Proof. Note that $I := \{k, \ldots, n\}$ is a maximal independent set in G (Observation 9b) and that the summation on the left hand side of Inequality (9) is over all the vertices in I. Hence $C := \{1, \ldots, k-1\}$ is a clique in G.

First we show that if $rc(G) \leq 2$, then the inequality is satisfied. Let $C_G : E(G) \to \{0, 1\}$ be a rainbow colouring of G. We can associate a codeword with each vertex $i \in I$ by reading the colours assigned by C_G to edges $\{i, c\}, c = 1, \ldots, d_i$. Since every pair

Algorithm 2 ColourThresholdGraph-Case1

Input: $G([n], E)$, a connected threshold graph, with $d_1 \geq \cdots \geq d_n$ and $\sum_{i=k}^{n} 2^{-d_i} \leq 1$, where d_i is the degree of vertex i and $k = \min\{i : 1 \leq i \leq n,\ d_i \leq i - 1\}$.

Output: A rainbow colouring $C_G : E(G) \to \{0, 1\}$ of G.

1: $I = \{k, \ldots, n\}$ // I is a maximal independent set in G
2: Let $\mathcal{B} = \{b_k, \ldots, b_n\}$ be a prefix-free code with $length(b_i) = d_i$ (constructed as mentioned in Observation 11)
3: **for** $i \in I$ **do**
4: $\quad C_G(\{i, j\}) = b_i(j),\ \forall j \in \{1, \ldots, d_i\}$
5: **end for**
6: **for** $i \in V(G) \setminus I$ **do** // $i < k$
7: $\quad C_G(\{i, j\}) = b_k(j),\ \forall j \in \{1, \ldots, i - 1\}$ // Note that $length(b_k) = d_k = k - 1$
8: **end for**
9: **return** C_G

$i, j \in I, d_i \leq d_j$ are non-adjacent, they need a 2-length rainbow path between them through a common neighbour $c \in \{1, \ldots, d_i\}$ (Observation 9c). This ensures that the codewords corresponding to i and j are complementary in at least one bit position. Hence the binary code formed by codewords corresponding to all the vertices in I form a prefix-free code. Hence the inequality is satisfied (by Theorem 10).

Conversely, if the inequality is satisfied, then Algorithm 2 gives a colouring C_G of $E(G)$ using at most 2 colours. We show that C_G is indeed a rainbow colouring of G. Consider any two non-adjacent vertices $i, j \in V(G)$, $i < j$. Since they are non-adjacent, either both of them are in I or otherwise j is in I and i is from the clique C such that $i > d_j$ (Since $N(j) = \{1, \ldots, d_j\}$). In the former case, $length(b_i) \geq length(b_j)$ and there exists a $v \in \{1, \ldots, d_j \leq d_i\}$ such that $b_j(v) \neq b_i(v)$ since b_j is not a prefix of b_i (They both belong to a prefix-free code B). Hence i–v–j is a rainbow path. Similarly in the latter case, $length(b_k) \geq length(b_j)$ and there exists a $v \in \{1, \ldots, d_j < i\}$ such that $b_j(v) \neq b_k(v)$ since b_j is not a prefix of b_k. Hence $C_G(\{v, j\}) \neq C_G(\{v, i\})$ and i–v–j is a rainbow path. Hence C_G is a rainbow colouring of G.

If G is not a clique, then $rc(G) \geq 2$ (Observation 1), and hence the above rainbow colouring is optimal. If G is a clique then $k = n$ and $|B| = |I| = 1$. So the single codeword b_n constructed as mentioned in Observation 11 has all the bits 0. So every edge of G is coloured using the single colour 0, which is optimal for G.

Since $\sum_{i=1}^{n} l_i = \sum_{i=1}^{n} d_i = 2m$, the prefix-free code B can be constructed in $O(m)$ time (Observation 11). Moreover, Algorithm 2 visits each edge only once. Hence the total time complexity is $O(m)$. □

Now we consider the case of threshold graphs which violate Inequality (9).

Theorem 13. *For every connected threshold graph G which does not satisfy Inequality (9),*

$$rc(G) = \max\{p, 3\},$$

where p is the number of pendant vertices in G.

Further, Algorithm 3 (ColourThresholdGraph-Case2) *gives an optimal rainbow colouring of G in $O(m)$ time*

Proof. It is easy to check, by considering all pairs of non-adjacent vertices, that Algorithm 3 indeed produces a rainbow colouring of G. It is also evident that it uses at most $\max\{p, 3\}$ colours. By Observation 2 and Theorem 12 , it follows hat $rc(G) \geq \max\{p, 3\}$.

Algorithm 3 ColourThresholdGraph-Case2

Input: $G([n], E)$, a connected threshold graph, with $d_1 \geq \cdots \geq d_n$, where d_i is the degree of vertex i.

Output: A rainbow colouring $C_G : E(G) \to \{0, \ldots, \max\{p, 3\} - 1\}$ of G, where p is the number of pendant vertices in G.

1: $P \leftarrow \{i \in V(G) : d_i = 1\}$, $p \leftarrow |P|$ // P is the set of pendant vertices in G
2: $C_G(\{p_i, 1\}) \leftarrow i - 1$ for each pendant vertex $p_1, \ldots, p_p$
3: **if** $p = n - 1$ **then**
4: **return** C_G // G is a star
5: **end if**
6: $C_G(\{1, 2\}) = 0$
7: **for** $i = 3$ **to** $i = n - p$ **do**
8: $C_G(\{i, 1\}) = 1$
9: $C_G(\{i, 2\}) = 2$ // Every $v \in \{3, \ldots, n - p\}$ is adjacent to vertices 1 and 2.
10: **end for**
11: $C_G(e) = 0$ for each edge e of G not coloured so far.
12: **return** C_G

Hence $rc(G) = \max\{p, 3\}$ and hence the rainbow colouring produced by Algorithm 3 is optimal. Further, since Algorithm 3 visits each edge only once, its time complexity is $O(m)$. □

Algorithm 4 ColourThresholdGraph

Input: $G([n], E)$, a connected threshold graph with $d_1 \geq \cdots \geq d_n$, where d_i is the degree of vertex i.

Output: An optimal rainbow colouring $C_G : E(G) \to \{0, \ldots, rc(G) - 1\}$ of G.

1: $k = \min\{i : 1 \leq i \leq n,\ d_i \leq i - 1\}$
2: **if** $\sum_{i=k}^{n} 2^{-d_i} \leq 1$ **then**
3: $C_G =$ ColourThresholdGraph-Case1(G)
4: **else**
5: $C_G =$ ColourThresholdGraph-Case2(G)
6: **end if**
7: **return** C_G

Combining Theorem 12 and Theorem 13, we get a complete characterisation for threshold graphs whose rainbow connection number is k, based on its degree sequence alone. Further we can find the optimally rainbow colour every threshold graph in linear-time.

Corollary 14. *Let $G([n], E)$, be a connected threshold graph with $d_1 \geq \cdots \geq d_n$, where d_i is the degree of vertex i. Then,*

$$rc(G) = \begin{cases} 1, & \text{if } G \text{ is a clique} \\ 2, & \text{if } G \text{ is not a clique and } \sum_{i=k}^{n} 2^{-d_i} \leq 1 \\ \max\{3, p\}, & \text{otherwise,} \end{cases} \tag{10}$$

where $k = \min\{i : 1 \leq i \leq n,\ d_i \leq i - 1\}$ and $p = |\{i : 1 \leq i \leq n,\ d_i = 1\}|$.

Further, Algorithm 4 (ColourThresholdGraph) *gives an optimal rainbow colouring of G in $O(m)$ time.*

References

[1] Prabhanjan Ananth, Meghana Nasre, and Kanthi K. Sarpatwar. Rainbow Connectivity: Hardness and Tractability. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, volume 13, pages 241–251, 2011.

[2] M. Basavaraju, L.S. Chandran, D. Rajendraprasad, and A. Ramaswamy. Rainbow connection number and radius. *Arxiv preprint arXiv:1011.0620v1*, 2010.

[3] M. Basavaraju, L.S. Chandran, D. Rajendraprasad, and A. Ramaswamy. Rainbow connection number of graph power and graph products. *Arxiv preprint arXiv:1104.4190*, 2011.

[4] Yair Caro, Arie Lev, Yehuda Roditty, Zsolt Tuza, and Raphael Yuster. On rainbow connection. *Electron. J. Combin.*, 15(1):Research paper 57, 13, 2008.

[5] Sourav Chakraborty, Eldar Fischer, Arie Matsliah, and Raphael Yuster. Hardness and algorithms for rainbow connection. *J. Comb. Optim.*, 21(3):330–347, 2011.

[6] G. Chartrand and P. Zhang. *Chromatic Graph Theory*. Chapman & Hall, 2008.

[7] Gary Chartrand, Garry L. Johns, Kathleen A. McKeon, and Ping Zhang. Rainbow connection in graphs. *Math. Bohem.*, 133(1):85–98, 2008.

[8] Thomas M. Cover and Joy A. Thomas. *Data Compression*, pages 103–158. John Wiley & Sons, Inc., 2005.

[9] A. Frieze and C.E. Tsourakakis. Rainbow connectivity of $g(n, p)$ at the connectivity threshold. *Arxiv preprint arXiv:1201.4603*, 2012.

[10] Peter L. Hammer and Bruno Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1981.

[11] Jing He and Hongyu Liang. On rainbow k-connectivity of random graphs. *Arxiv preprint arXiv:1012.1942v1 [math.CO]*, 2010.

[12] Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981.

[13] S. Khot. Hardness results for coloring 3-colorable 3-uniform hypergraphs. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 23–32. IEEE, 2002.

[14] L.G. Kraft. A device for quanitizing, grouping and coding amplitude modulated pulses. Master's thesis, Electrical Engineering Department, Massachusetts Institute of Technology, 1949.

[15] Michael Krivelevich and Raphael Yuster. The rainbow connection of a graph is (at most) reciprocal to its minimum degree. *J. Graph Theory*, 63(3):185–191, 2010.

[16] S. Li and X. Li. Note on the complexity of determining the rainbow connectedness for bipartite graphs. *Arxiv preprint arXiv:1109.5534*, 2011.

[17] X. Li and Y. Sun. *Rainbow Connections of Graphs*. Springerbriefs in Mathematics. Springer, 2012.

[18] Xueliang Li and Yuefang Sun. Rainbow connections of graphs – a survey. *Arxiv preprint arXiv:1101.5747v2 [math.CO]*, 2011.

[19] Xueliang Li and Yuefang Sun. Upper bounds for the rainbow connection numbers of line graphs. *Graphs and Combinatorics*, pages 1–13, 2011. 10.1007/s00373-011-1034-1.

[20] J. Misra and David Gries. A constructive proof of vizing's theorem. *Information Processing Letters*, 41(3):131 – 133, 1992.

[21] Ingo Schiermeyer. Rainbow connection in graphs with minimum degree three. In *Combinatorial Algorithms*, volume 5874 of *Lecture Notes in Comput. Sci.*, pages 432–437. Springer, Berlin, 2009.

[22] Harold. Seward, H. Information sorting in the application of electronic digital computers to business operations. Master's thesis, Digital Computer Laboratory, Massachusetts Institute of Technology, 1954.

[23] Yilun Shang. A sharp threshold for rainbow connection of random bipartite graphs. *Int. J. Appl. Math.*, 24(1):149–153, 2011.

[24] L. Sunil Chandran, Anita Das, Deepak Rajendraprasad, and Nithin M. Varma. Rainbow connection number and connected dominating sets. *Journal of Graph Theory*, 2011.

[25] A. Wigderson. The complexity of graph connectivity. *Mathematical Foundations of Computer Science 1992*, pages 112–132, 1992.