The University of Resources. Since 1765.

Introduction to the MINIMUM RAINBOW SUBGRAPH problem

By the Faculty of Mathematics and Computer Sciences
of the Technische Universität Bergakademie Freiberg

approved

Thesis

to attain the academic degree of

doctor rerum naturalium
(Dr. rer. nat.)

submitted by **Dipl.-Math. Stephan Matos Camacho**

born on the 18th November 1980 in Bad Muskau

Assessor: Prof. Dr. rer. nat. Ingo Schiermeyer
Prof. Dr. rer. nat. Hubert Randerath

Date of the award: Freiberg, 13th March 2012

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Dr. Ingo Schiermeyer for his permanent support and guidance. Without his numerous hints and his feedback this thesis would not have been achieved.

Furthermore special thanks go to Dr. Anja Kohl for an almost final view on this thesis and my colleagues for their faith where mine flagged.

Finally I want to thank God for his guidance and all the gifts he gave to me, the ways he opened and the shelter I found in him.

Mathematics and biology - having nothing in common?

When I started my studies of mathematics I very soon realized, that one of the biggest advantages of a mathematician is being able to map the real world into a formal language, a set of formulas or an abstract description. Almost as soon as the first observation I found out, that tackling problems in this structured and well ordered world is much more interesting and quite often a lot of more fun. Unfortunately, sometimes the accomplished solution turned out not being very feasible or of practical relevance. For example, a mathematician working on graph theory would be able to construct an algorithm to solve the frequency assignment problem exactly, but using it in the real would take much too long time in contrast to the goal of assuring connections between two mobile phones in almost no time. Thus, the companies trust on fast heuristics instead of resource preserving exact solutions.

Nevertheless, graph theory is often motivated by applied problems as the already mentioned frequency assignment problem, the "Seven Bridges of Königsberg" or the Travelling Salesman problem. Regrettable, a lot of these problems, some may argue only the interesting ones, turn out to be only hard solvable, or in terms of computer science, are NP-hard. For example, if we try to handle the metric version of the Travelling Salesman problem, the best know algorithm, talking about the approximation ratio, is the one from Nico Christofides ([4]). Unless there are a lot of new heuristics found in the mean time, no one was able to guarantee a tour shorter than $\frac{3}{2}$ of the optimal length. Nonetheless, algorithms were implemented that solve the problem on a given set of cities up to a constant close to 1, but failing on an arbitrary collection.

The problem this thesis is introducing, the so-called Minimum Rainbow Subgraph problem, is one of these hard ones, too. Originally it was a biological question ([22]), which was discussed by

computer scientists first ([1], [7], [10] or [14]). We translated it into the language of graph theory and ended with a model, which also covers general graphs: the Minimum Rainbow Subgraph problem.

This thesis shall be seen as an introduction to the topic and will provide, beside basic definitions and preliminary observations, first algorithmic attempts to tackle minimum rainbow subgraphs. Due to its hardness we will mainly discuss heuristics and their performance compared to exact solutions. In the last chapter some more details on graphs, arisen from biological instances, are given in order to provide starting points for future work. Maybe these or other special properties of such graphs can be used to develop algorithms working more efficient and calculating solutions closer to the minimum on real world instances.

Let's start with some more information on the biological background in order to disprove the well-known *Theorem of Schaar*: "There is no application of (my) graph theory!"

Part I.

Going for a start

1. Introducing haplotyping

With the completion of the Human Genome Project in 2003 the structure of the human genome is fully decrypted and known to us. But, unfortunately, almost no two individuals share the same sequence of nucleotides. 99% of the positions are identical, the rest may differ ([22]). About 90% of these genetic variations, so-called polymorphisms, are *Single Nucleotide Polymorphisms* (SNPs), which are a variation in only one site of the human genome. Consequentially, their importance can hardly be overestimated, and there is a wide field of therapeutic, diagnostic and forensic applications. Furthermore, since these SNPs can be inherited, there is a good chance that they play a big role in the pathogenesis of diseases as cancer, cardiovascular disease, diabetes, psychiatric illnesses, autoimmunity and others. Maybe identifying the positions of SNPs could lead to improved prevention, diagnosis and treatment of diseases ([16]).

Humans are *diploid* organisms, i.e., their DNA is organized in pairs of chromosomes. Every one of these pairs consists of one chromosome copy inherited from the mother, the maternal one, and another copy inherited from the father, the fraternal one. For a given SNP, we call an individual *homozygous* if the two copies do not differ in this position, or *heterozygous* if they differ. A *haplotype* is defined by the values of a set of SNPs on a given chromosome copy, i.e., the fraternal or the maternal one. When we refer to a *genotype*, we will talk about both of the realisations in the two chromosomes. So, *haplotyping* is the determination of a set of haplotypes explaining a given set of genotypes. By explaining a genotype g we mean that there is a pair h_1 and h_2 of haplotypes giving the required genome, in reference to the given chromosomes. In short, we will denote it by

$$g = h_1 \oplus h_2.$$

In the following we consider genomes as a collection of strings, or sequences consisting of entries

from the set $\{A, G, C, T\}$. Thereby every letter encodes one of the four possible nucleotides in the human genome, i.e., adenine, guanine, cytosine and thymine.

Example 1.1. Consider the following pair of chromosomes c_1 and c_2 with

$$c_1 : \texttt{taggtccCtatttCccaggcgcCgtatacttcgacgggTctata}$$
$$c_2 : \texttt{taggtccGtatttAccaggcgcGgtatacttcgacgggTctata}.$$

The corresponding haplotypes are $h_1 = CCCT$ and $h_2 = GAGT$. In fact, there is only a variation in the first, second and third SNP, the forth site is identical in both of the chromosomes.

Remark. Do not get confused by the DNA being itself organised pairwise in the well-known double helix structure. There adenine is always paired with thymine, and guanine with cytosine. Think of a pair of strings, where a position in one string always depends on the corresponding site in the other one. We will only refer to one side of these pairs, i.e., one string of the double helix. Thus, a SNP is not a discrepancy inside a single pair of a given DNA string, e.g., cytosine as counterpart of adenine instead of thymine. It is a variance of the whole pair at one position in the examined DNA string with respect to the referred half, e.g., there is the pair adenine-thymine instead of thymine-adenine.

If we talk about the general *Population Haplotyping problem* (PHP), we have a given set $\mathcal{G}$ of genotypes we are trying to explain with a set $\mathcal{H}$ of haplotypes, such that for every genotype $g \in \mathcal{G}$ there exist two haplotypes h' and h'' explaining g, i.e., $g = h' \oplus h''$. There are several approaches to solve the PHP. We will present three very common ones, taken from [22]:

Perfect Phylogeny In this approach we are looking for a so-called *perfect phylogeny* solution, in other words, our obtained haplotype set $\mathcal{H}$ must fit in a tree T, such that

(i) the haplotypes are the leaves of T

(ii) every SNP s is represented by an edge e_s in T

(iii) deleting an edge e_s leads to a partition of $\mathcal{H}$, where haplotypes of the same partition set have the same nucleotides at position s.

Solving PHP under this condition can be done in polynomial time (see [1], [7]).

Clark's rule This greedy rule starts from a minimal initial set of haplotypes, e.g., genotypes with no 2-entries, and tries to resolve as many as possible genotypes. It introduces new haplotypes only when needed ([5]). Gusfield showed that this approach is APX-hard and formulated an Integer Programming problem for its solution ([9], [10]).

Pure Parsimony Here the foundation is, that we try to solve our problem with as few as possible elements, meaning we are trying to determine a set $\mathcal{H}$ of haplotypes of minimum cardinality to explain the given set $\mathcal{G}$ of genotypes. Gusfield studied this problem ([11]), too, and adapted an Integer Programming formulation as a practical solution. Nevertheless, this problem is NP-hard, as shown by Hubbel ([15]).

There are a lot of other models and variations, too. Due to the intention of this paper we will therefore refer to the nice overview provided by Bonizzoni et al. ([3]).

The main approach in this thesis is pure parsimony.

2. Becoming mathematical

Fortunately, for most of the known SNPs we only have two possible nucleotides, called *alleles*, occurring there. Therefore it is feasible to denote haplotypes by vectors in $\{0,1\}$, and a genotype g by a vector in $\{0,1,2\}$, with a 2-entry if and only if the referred SNP varies in the two corresponding haplotypes. Each position where a 2 appears is called *ambiguous* position. Continuing Example 1.1 we could write

$$h_1 = 0000,\ h_2 = 1110,\ g = h_1 \oplus h_2 = 2220.$$

Then our haplotype addition $g = h' \oplus h''$ for given haplotypes h', h'' and a genotype g can be defined as follows

$$g[i] := \begin{cases} 0 & \text{if } h'[i] = h''[i] = 0 \\ 1 & \text{if } h'[i] = h''[i] = 1 \\ 2 & \text{if } h'[i] \neq h''[i], \end{cases}$$

if $h[i]$ describes the ith position of the vector h.

For the *Pure Parsimony Haplotyping problem* (PPH problem) we may discuss several classes, e.g., we may classify given problem instances by the number of ambiguous sites in the genotypes. Assume, we have an instance consisting of p different genotypes $g_1, \ldots, g_p$ out of a population. Then we denote the corresponding PPH problem by PPH(k) if each genotype has at most k ambiguous sites.

There are some trivial bounds on $|\mathcal{H}|$ ([22]):

Fact 2.1. *Given a set $\mathcal{G}$ of p genotypes, there always exists an $\mathcal{H}$ with $|\mathcal{H}| \leqslant 2p = 2|\mathcal{G}|$.*

Proof. For a genotype $g \in \mathcal{G}$ let h' be the haplotype having a 0 whenever g has one, and a 1 at all other positions. Respectively, let h'' be the haplotype having a 1 whenever g has one, and a 0 at all other positions. Then $g = h' \oplus h''$. In this way we can find a pair of haplotypes of every given genotype g. q.e.d.

Fact 2.2. *Let $\mathcal{H}$ be a set explaining a given set $\mathcal{G}$ of p genotypes. Then $|\mathcal{H}| > \sqrt{2p}$.*

Proof. Assuming $\mathcal{H}$ explains $\mathcal{G}$, then every $g \in \mathcal{G}$ can be associated with a different pair of haplotypes out of $\mathcal{H}$. Hence, $p = |\mathcal{G}| \leqslant \binom{|\mathcal{H}|}{2} = \frac{|\mathcal{H}|(|\mathcal{H}|-1)}{2} < \frac{|\mathcal{H}|^2}{2}$. q.e.d.

In fact, the lower bound can be given more precisely by

$$|\mathcal{H}| \geqslant \frac{1+\sqrt{1+8p}}{2}.$$

For further illustration, we will extend the first example:

Example 2.1. The set $\mathcal{G}$ consists of four genotypes:

$$\mathcal{G} = \{g_1 = 2022, g_2 = 2021, g_3 = 2220, g_4 = 2222\}.$$

Then you may explain $\mathcal{G}$ with

$$g_1 = 0000 \oplus 1011, \qquad g_2 = 0011 \oplus 1001,$$
$$g_3 = 1000 \oplus 0110, \qquad g_4 = 1010 \oplus 0101,$$

using eight different haplotypes.

But a possible set of minimum order might be

$$\mathcal{H} = \{h_1 = 0001, h_2 = 0100, h_3 = 1010, h_4 = 1011\},$$

with $g_1 = h_1 \oplus h_3$, $g_2 = h_1 \oplus h_4$, $g_3 = h_2 \oplus h_3$ and $g_4 = h_2 \oplus h_4$.

Hubbel ([15]) discussed the complexity of the Pure Parsimony Haplotyping problem and showed its NP-hardness. Later Lancia et al. extended this result in [22]:

Theorem 2.1 ([22]). *The PPH(k) problem is NP-hard for $k \geqslant 3$. In this case it is even APX-hard, that is, there is a constant $c > 1$ such that the existence of a c-approximation for this problem would imply P=NP.*

The remaining cases are more convenient to handle.

Theorem 2.2 ([23]). *The PPH(k) problem is solvable in polynomial time for $k = 0, 1, 2$.*

Lancia and Rizzi used a reduction to the Vertex Cover problem, to show the APX-hardness, and a Linear Integer Programming formulation for the NP-hardness. Moreover, they stated an exact algorithm using again Vertex Cover for the polynomial case. Since they were able to reduce this problem to searching for a minimum vertex cover in a bipartite graph, which can be done in polynomial time, their conclusion followed naturally. In addition, they have given some approximation results:

Theorem 2.3 ([22]). *(i) The PPH problem can be approximated by a $\sqrt{p}$-approximation algorithm in polynomial time for a given set $\mathcal{G}$ of p genotypes.*

(ii) There exists a 2^{k-1}-approximation algorithm for the PPH(k) problem.

Remark. Due to the König-Egerváry Theorem ([20]) it is well known that Vertex Cover and Maximum Matching are equivalent in bipartite graphs. On the other hand, Maximum Matching can be done in polynomial time (see [6]). There are some fast and efficient algorithms for bipartite graphs, e.g., in [13], that can be used for solving Vertex Cover as well.

In the literature you can find a different name for the PPH problem. For example, in [14] it is denoted by *Optimal Haplotype Inference (OHI) problem*. There, Huang et al. suggested an iterative semidefinite programming-based approximation algorithm, called SDPHapInfer. They were able to show that, with a high probability, the algorithm SDPHapInfer finds a solution of $\mathcal{O}(\log p)$ approximation within $\mathcal{O}(\log p)$ steps of iteration.

Of course, several other approaches were discussed, e.g., genetic algorithms ([27]) or statistical methods like Maximum-Likelihood estimations ([8]) were studied.

In the next chapter we will transform the PPH problem in a graph theoretical one. Starting with some basic concepts, we will present known models leading to the Minimum Rainbow Subgraph problem.

Part II.

The MRS problem

3. The graph theoretical point of view

In this chapter we will use graph theory to model the PPH problem. Since graphs are a very powerful and vivid tool, maybe we have a better chance to receive good heuristics or to find some special cases, which can be solved in polynomial time. First, we have to translate the PPH problem into the language of vertices and edges (for a more sufficient overview on basic definitions and concepts, have a look at [28]):

A *graph* $G = (V, E)$ consists of two sets, where V is the set of *vertices* and E is the set of *edges* of G. Thereby, an edge is defined by its two end vertices. So an edge e is $e = \{x, y\}$ with $x, y \in V$. I.e., the vertex set E is $E \subseteq \mathcal{V}^{[2]}$, where $\mathcal{V}^{[2]}$ is the set of all two-element subsets of V. In short we often refer e by its vertices, say, $e = xy$. The number of vertices, the *order* of G, will be denoted by $n = n(G)$, the number of edges, the *size* of G, by $m = m(G)$. For our considerations, n and m will be finite.

The vertices $x, y \in V$ are called *adjacent*, if there is an edge $e = xy \in E$. Similarly we define adjacency for edges $e, f \in E$, if there exists a vertex $x \in V$ being end vertex of both edges. In this case x and e, respectively x and f, are *incident*. The set of vertices adjacent to a vertex x is called its *neighbourhood* $N(x)$, respectively, the *neighbourhood* $N(e)$ of an edge e is the set of all adjacent edges. Furthermore, we refer to the closed neighbourhood $N[v]$, if v is included.

For the vertex $x \in V$ we define by the *degree* or *valency* the number of edges incident to x. In short we will write $d(x)$. Using this we can define the maximum (minimum) degree $\Delta(G)$ ($\delta(G)$) of the graph G by

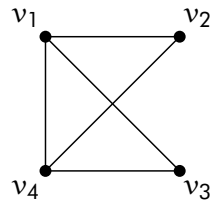
$$\Delta(G) = \max\{d(x) : x \in V\}, \ \delta(G) = \min\{d(x) : x \in V\}.$$

Often not the whole graph G itself is of interest, but we only want to consider certain vertices and edges of it. Then we talk about a *subgraph* H of G, if H is a graph with $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We call the subgraph H *induced* if $E(H)$ consists of all edges of G between the vertices of H, in symbol $G[H]$.

Furthermore we want to colour the edges of the graph. Therefore we define an edge colouring c as a mapping $c : E \to \mathcal{C}$, where $\mathcal{C}$ will be a given set of colours. Often for convenience we take $\mathcal{C} = \{1, 2, \ldots, k\}$ as colour set. We call an edge colouring c proper, if no two adjacent edges are coloured identically.

Now it is time to develop a first model for the PPH problem. Sharan et al. in [26] used so-called *Clark-consistency graphs*. For a given PPH instance on p genotypes in $\mathcal{G}$ every vertex $x \in V(G)$ of the consistency graph G corresponds to a genotype. Two vertices $x, y \in V(G)$ are adjacent if there exists a haplotype compatible with both referred genotypes. At this juncture a haplotype h being *compatible* to the genotype g means that there exists another haplotype $\widetilde{h}$, such that $g = h \oplus \widetilde{h}$. Continuing Example 2.1, Figure 3.1 shows the corresponding Clark-consistency graph.

Fig. 3.1.: Example 2.1



Obviously, this kind of graph is inspired by Clark's Rule. In the paper ([26]), also (k, l)-*bounded instances* are defined. Here k symbolises a limitation on the number of 2-entries in the genotypes, l a limitation on the number of genotypes, that have a 2 at the same position. Apart from that, we can use an asterisk instead of k or l to refer to no limitation. Using this formulation the PPH(k) problem can be seen as a $(k, *)$-bounded problem.

Sharan et al. proved several results on special bounded instances to show the hardness of even restrictive problems:

Theorem 3.1 ([26]). *Parsimony haplotyping for $(4,3)$-bounded instances is NP-hard, and even APX-hard.*

Even more, an apparently weaker formulation was proven not being much easier:

Theorem 3.2 ([26]). *Let MHC be the problem of finding a minimum set of haplotypes $\mathcal{H}$ to a given set of genotypes, such that each genotype is consistent with some haplotype in $\mathcal{H}$. Then MHC is NP-complete.*

But they also found some positive results:

Theorem 3.3 ([26]). *In respect to the number of the occurring haplotypes in the solution set parsimony haplotyping is fixed-parameter tractable.*

Later on, they studied complete Clark-consistency graphs, meaning that in the corresponding set of genotypes every two genotypes share at least one compatible haplotype. In general, this restriction leads to no improvement, the PPH problem on so-called clique instances is still NP-hard. But they were able to state a polynomial time algorithm to solve this problem on $(*,2)$-bounded clique instances. Furthermore, the existence of a polynomial time algorithm for $(\mathcal{O}(\log n),*)$-bounded instances if the consistency graph has bounded treewidth was shown, as well as the NP- and APX-hardness for problems leading to a bipartite consistency graph.

Our approach will change the roles of genotypes and haplotypes in the corresponding graph and adds the aspect of colouring graphs to the model.

3.1. The MRS problem

Let $\mathcal{G} = \{g_1, \ldots, g_p\}$ be a given set of genotypes. Then for every genotype $g \in \mathcal{G}$ with k ambiguous sites there are 2^{k-1} pairs of haplotypes explaining it. We construct our graph G as follows:

(i) For every genotype $g_i \in \mathcal{G}$ ($i \in \{1, \ldots, p\}$) find the explaining haplotypes h_{ij} ($j \in \mathbb{N}$).

(ii) Identify the haplotypes with the vertices of G.

(iii) Two vertices are adjacent if and only if the two corresponding haplotypes explain a genotype $g_i \in \mathcal{G}$. Colour the edge with the colour $i \in \{1, \ldots, p\}$.

Remark. For genotypes having no 2-entry, there is only one edge in this colour. Furthermore, this edge is a loop, meaning starting and ending in the same vertex.

Remark. If G contains no loops, then this edge colouring is proper, i.e., there are no two adjacent edges of the same colour.

Obviously, for solving the PPH problem, we are looking for a subgraph of G with p pairwise distinct coloured edges on the minimum number of vertices.

Remark. If G is an edge coloured graph, than a subgraph H of G is called rainbow, if all edges of H are coloured distinct.

Definition 3.1. Let $G = (V, E)$ be a graph and $c : E \to \{1, \ldots, p\}$ an edge colouring of G. Then the *Minimum Rainbow Subgraph problem* (MRS problem) consists of finding a subgraph $F \subseteq G$ of minimum order with p edges, such that every colour appears exactly once. The order of an optimal solution F will be denoted by $\mathrm{rs}(G) = |V(F)|$.

Remark. Every subset $\mathcal{H}$ explaining $\mathcal{G}$ corresponds to a rainbow subgraph H of G

For some analysis it might be useful to consider simple graphs, i.e., graphs without loops or multiple edges. (Of course, multiple edges will not occur in graphs arisen from a given set $\mathcal{G}$ of genotypes). On the other side, loops will only occur if the corresponding genotype has no 2-entry. Hence, there is only one possibility to explain this genotype, namely by adding it to $\mathcal{H}$. It is obvious, that there will be exactly one edge of the appropriate colour in the graph G and the vertex of this edge must be in any rainbow subgraph F. Starting with the graph G, we will construct a graph G^* as follows:

(i) Let $V(G^*) = V(G_1) \cup V(G_2)$, where G_1 and G_2 are two copies of G.

(ii) Let $V(G_1) = \{v_1, \ldots, v_n\}$ and $V(G_2) = \{w_1, \ldots, w_n\}$ be the vertices of the copies and $\{v_1, \ldots, v_t\}$ and $\{w_1, \ldots, w_t\}$ ($t \leqslant n$) are the $2t$ vertices having a loop. Delete these loops and add edges $v_1w_1, v_2w_2, \ldots, v_tw_t$ to G^*. Colour each edge v_iw_i, for $i \in \{1, \ldots, t\}$, with the same colour the deleted loop of the two vertices was coloured before.

(iii) Let $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$ be two edges coloured with the same colour i, where $i \in \{1, \ldots, p\}$. Then recolour e_2 with the colour $p + i$.

Remark. The new loop-free graph G^* is coloured with $2p - t$ colours and $\Delta(G^*) \leqslant \Delta(G)$ holds.

This transformation does not change anything concerning the MRS approximation:

Lemma 3.1. *If an algorithm A guarantees a performance of $1+\epsilon$ for the MRS problem for graphs without loops, so it does for graphs with loops.*

Proof. Let G be a graph with loops and G^* is constructed as given before. Furthermore, let H^* be a rainbow subgraph of G^* obtained by algorithm A, F^* the optimal solution for G^*, respectively F for G.

We define $V(H_i) := V(H^*) \cap V(G_i)$, $i = 1, 2$. Without loss of generality let

$$\min\{|V(H_1)|, |V(H_2)|\} = |V(H_1)|.$$

Obviously, $|V(F^*)| = 2|V(F)|$ and

$$\frac{|V(H^*)|}{|V(F^*)|} = \frac{|V(H_1)| + |V(H_2)|}{2|V(F)|} \geqslant \frac{\min\{|V(H_1)|, |V(H_2)|\}}{|V(F)|} = \frac{|V(H_1)|}{|V(F)|}.$$

q.e.d.

In the following, we will discuss the MRS problem only for graphs without loops.

As seen before, any instance of the PPH(k) problem can be reduced into a MRS problem instance in polynomial time. Hereby, every haplotype is represented by a vertex in the corresponding graph G. Furthermore, $|\mathcal{H}| = |V(F)|$ for every set $\mathcal{H}$ of haplotypes of minimum cardinality and for every minimum rainbow subgraph F of G. Talking structurally, there exists a bijection from the sets $\mathcal{H}$ explaining the set of genotypes $\mathcal{G}$ onto the minimum rainbow subgraphs F of G. This implies that any approximation algorithm with arbitrarily good performance guarantee for the MRS problem would lead to an approximation algorithm with arbitrarily good performance for the PPH(k) problem, a contradiction. Together with the MRS problem obviously lying in NP, we can formulate the following theorem:

Theorem 3.4 ([24]). *The Minimum Rainbow Subgraph problem is NP-hard, and even APX-hard.*

In [17] Katrenič and Schiermeyer took a closer look on this and were able to prove the following theorem:

Theorem 3.5 ([17]). *For graphs with maximum degree $\Delta = 2$ the MRS problem remains NP-hard.*

The authors proved it by reducing 3-OCC-MAX 2SAT to MRS problem instances with maximum degree $\Delta = 2$. Here 3-OCC-MAX 2SAT is the restriction of the general maximum satisfiability problem to instances containing only clauses with at most two variables, where each variable occurs in at most three clauses. This was shown by Berman and Karpinksi ([2]) being NP-hard, too.

Starting with an instance F of a 3-OCC-MAX 2SAT problem consisting of c clauses $C_1, \dots, C_c$, they constructed a graph of the $4n + c$ vertices $a_1, \dots, a_c, x_1^1, \dots, x_1^n, x_1^2, \dots, x_n^2, \ \bar{x}_1^1, \dots, \bar{x}_n^1$, $\bar{x}_1^2, \dots, \bar{x}_n^2$. Furthermore, they used $n + c$ colours, $A_1, \dots, A_c$ and $B_1 \dots, B_n$ for the following edges: $x_i^1 x_i^2$ and $\bar{x}_i^1 \bar{x}_i^2$ are coloured by B_i, where $1 \leqslant i \leqslant n$, and for each literal l the edge $l^{occ(l,j)} a_j$ is coloured by A_j. Hereby, $occ(l, j)$ represents the number of occurrences of the literal l in the clauses $C_1, \dots, C_j$. It is easy to see that the resulting graph has maximum degree at most 2. It remains to show that at least s clauses can be satisfied in F if and only if the MRS for the graph has at most $2n + 2c - s$ vertices.

First assume that R is a solution for F satisfying s clauses. Then a solution H for the corresponding graph is given by the following $2n+2c-s$ vertices: take all vertices $a_1 \dots, a_c$, and for each literal $l \in R$ we use the vertices $\{l^1, l^2\}$. Apparently, at most $c - s$ colours of the colours $A_1, \dots, A_c$ are not covered by H, but then add a vertex l^1, $l \in C$, for each clause C unsatisfied in R.

On the other site, if we have a MRS H with $2n + 2c - s$ vertices, we get a solution R for F as follows: For covering colour B_i, where $i \in \{1, \dots, n\}$, one of the pairs $\{x_i^1, x_i^2\}$ and $\{\bar{x}_i^1, \bar{x}_i^2\}$ has to be in H. Therefore, set x true, if $\{x^1, x^2\}$ is in H, otherwise put $\bar{x}$ into R. In addition, H must contain all vertices $a_1, \dots, a_c$ to cover the colours $A_1, \dots, A_c$. Now consider the subgraph $H' \subseteq H$, where $V(H') = \{a_1, \dots, a_c\} \cup \{l^q : l \in R, q = 1, 2\}$. Then $|H'| = c + 2n$ and does not cover at most $c - s$ colours of $A_1, \dots, A_c$. But for every colour A_j, with $i \in \{1, \dots, c\}$,

corresponding to the satisfaction of clause C_j, R still satisfies at least s clauses. This concludes the proof.

Apart from that, Katrenič and Schiermeyer presented a deterministic algorithm, which solves the MRS problem exactly in $\mathcal{O}(2^{(p+2p\log_2\Delta)}n^{\mathcal{O}(1)})$ time and $\mathcal{O}(2^p n^{\mathcal{O}(1)})$ space.

In [24] some simple bounds to classify the range of an optimal solution are given.

Lemma 3.2 ([24]). *Let $G=(V,E)$ be a graph with maximum degree $\Delta(G)=\Delta$, whose edges are properly coloured with p colours. Then*

$$\frac{2p}{\Delta} \leqslant rs(G) \leqslant 2p.$$

Proof. The upper bound of $2p$ can be easily realised by taking p distinct edges, every edge in a different colour. The lower bound arises from the Handshaking Lemma and taking only vertices of maximum degree Δ in the solution. q.e.d.

A tight lower bound may be of major importance for the design of approximation algorithms in order to provide better approximation ratios. Therefore, we are looking for improvements of Lemma 3.2. Introducing the *colour degree* $cd(v)$, that denotes the number of distinct colours among all edges incident to a vertex $v \in V$, and let $cd(i) = \max\{\, cd(v) \mid v \in V \text{ has an incident edge in colour } i\}$ be the *maximum colour degree* for every colour i, where $i \in \{1,\dots,p\}$, we formulate the following corollary:

Corollary 3.3. *Let $G=(V,E)$ be a p-edge-coloured graph. Then*

$$rs(G) \geqslant \sum_{i=1}^{p} \frac{2}{cd(i)}.$$

Proof. Let $F \subseteq G$ be a minimum rainbow subgraph of G. Then

$$rs(G) = \sum_{i=1}^{|V(F)|} 1 = \sum_{\substack{e\in E(F),\\ e=uv}} \frac{1}{d(u)+d(v)} \geqslant \sum_{i=1}^{p} \frac{2}{cd(i)}.$$

q.e.d.

Obviously, $\sum_{i=1}^{p} \frac{2}{cd(i)} \geqslant \frac{2p}{\Delta}$ if the maximum degree of the graph G is denoted by Δ. Let us have a look at the following example:

Example 3.1. [19] Consider for $p \geqslant 4$ and $\Delta \geqslant 2$ the graph $G = K_{1,\Delta} + C_{p-1}$, where G is the disjoint union of a star of order $\Delta + 1$ and a cycle of length $p - 1$. Furthermore let all edges of the cycle C_{p-1} be coloured distinct, e.g., with the colours $1, \ldots, p-1$, and all edges of the $K_{1,\Delta}$ are coloured with colour p. Then

$$rs(G) = p + 1 = p - 1 + 2 = \sum_{i=1}^{p} \frac{2}{cd(i)} > \frac{2p}{\Delta}.$$

There is a further improvement on this bound if we count the number of distinct colours of the edges adjacent to an edge $e \in E$. We define

$$q(i) = \min \left\{ \frac{1}{d(u)} + \frac{1}{d(v)} \ : \ uv = e \in E \text{ and } c(e) = i \right\}.$$

Corollary 3.4. *Let $G = (V, E)$ be a p-edge-coloured graph with maximum degree $\Delta(G) = \Delta$. Then*

$$rs(G) \geqslant \sum_{i=1}^{p} q(i) \geqslant \sum_{i=1}^{p} \frac{2}{cd(i)} \geqslant \frac{2p}{\Delta}.$$

Example 3.2. [19] Let $G \cong K_{1,p}$ for some integer $p \geqslant 1$, where all edges are coloured distinct. Then $q(i) = 1 + \frac{1}{p}$ and

$$rs(G) = p + 1 = p \cdot \left(1 + \frac{1}{p}\right) = \sum_{i=1}^{p} q(i) > 2 = p \cdot \frac{2}{p} = \sum_{i=1}^{p} \frac{2}{cd(i)} = \frac{2p}{\Delta(G)}.$$

3.2. The MRS problem on special graph classes

Since we are only discussing graphs without loops, the maximum degree of our graphs is bounded from above by p, the number of colours. On the other hand, using vertices of maximum degree in the optimal solution F seems to be a good heuristic to minimize the order of F. In her diploma thesis ([18]), Maria Koch showed some interesting results on special graph classes and presented a greedy heuristic to get an approximating solution for the MRS problem.

We will start with a simple upper bound depending on the maximum degree $\Delta(G)$:

Theorem 3.6 ([18]). *Let $G = (V, E)$ be a loop-free p-edge-coloured graph with maximum degree $\Delta(G)$. Then for the optimal solution of the MRS problem $F \subseteq G$ the following inequality holds:*

$$|V(F)| \leqslant 2p + 1 - \Delta(G).$$

This bound is not difficult to discover, since we have seen that the worst case in explaining p genotypes is using two distinct haplotypes for each. If there is a vertex of maximum degree $\Delta(G)$ in G, exactly $\Delta(G)$ edges in pairwise distinct colours are incident to it. If we use it and its neighbours for explaining the genotypes, only $p - \Delta(G)$ colours are left, needed to be added. The worst case is again using two distinct vertices for each colour, therefore we have

$$|V(F)| \leqslant \Delta(G) + 1 + 2(p - \Delta(G)) = 2p + 1 - \Delta(G).$$

Furthermore, this result leads to a nice characterisation of graphs realising this bound:

Theorem 3.7 ([19]). *Let $G = (V, E)$ be a properly p-edge-coloured graph with maximum degree $\Delta(G) = \Delta$. Then $rs(G) = 2p + 1 - \Delta$ leads to the following properties:*

(i) *G contains a star $K_{1,\Delta}$, where v_0 is the center vertex and $v_1, v_2, \ldots, v_\Delta$ are its leaves. Moreover, $G[N(v_0)]$ is independent. Let $c(v_0, v_i) = i$ for $1 \leqslant i \leqslant \Delta$ and $H_0 \cong G[N[v_0]]$.*

(ii) *For $p > \Delta(G)$ there are pairwise vertex-disjoint subgraphs $H_{\Delta+1}, H_{\Delta+2}, \ldots H_p$, where H_i is spanned by the edges with colour i and $V(H_0) \cap V(H_i) = \emptyset$ for $\Delta + 1 \leqslant i \leqslant p$.*

(iii) *$E(H_i, H_j) = \emptyset$ for $\Delta + 1 \leqslant i < j \leqslant p$.*

(iv) *$E(v_i, H_j) = \emptyset$ for $1 \leqslant i \leqslant \Delta$ and $\Delta + 1 \leqslant j \leqslant p$.*

In an almost similar way we can derive the following upper bound for r-regular graphs, where $\delta(G) = \Delta(G) = r$:

Theorem 3.8 ([18]). *Let $G = (V, E)$ be a loop-free p-edge-coloured r-regular graph on n vertices. Then for the optimal solution of the MRS problem $F \subset G$ the following inequality holds:*

$$|V(F)| \leqslant 2p - r - s + 2,$$

where $s = \left\lceil \frac{p-r}{n-1} \right\rceil$.

Here, any arbitrary vertex v_0 and its r neighbours add r colours to the solution. By construction and according to the Pigeon-Hole Principle, there is a vertex $v_1 \in V - \{v_0\}$, remember $|V - \{v_0\}| = n - 1$, with at least s incident edges of pairwise distinct colours, which can be added to the solution, too. For the rest we use the "brute force" approach by taking two vertices for each colour.

If we take a closer look on paths, then unfortunately even the simple structure of $n-2$ vertices of degree 2 and two vertices of degree 1 does not lead to tighter bounds. The closest we can get is if there are many edges of the same colour. Then we can use the Pigeon-Hole Principle again and guarantee, that we could leave a vertex out. We are in the situation that at least two edges of colours occurring twice or more often share a vertex. This is the one that will be left out.

Theorem 3.9 ([18]). *Let* $G = (V, E)$ *be a connected,* p*-edge-coloured path of length* m*,* $s = m+p$*,* $s \geqslant 2$ *and* k *the number of colours occurring at least twice in* G*. Then*

$$rs(G) \leqslant p + s + t,$$

with

$$t = \begin{cases} 0 & \text{, if } s + k > p - k, \\ 1 & \text{, else.} \end{cases}$$

4. Trying to be not that bad

We have already discovered that the MRS problem is NP-hard and even APX-hard. Nevertheless, our main task remains providing some algorithms, which solve the MRS problem for a given instance in quite a reasonable time. Therefore we are forced to explore approximating algorithms for the question "NP=P?" unlikely being answered for the next years.

A first approximation result arises from Lemma 3.2:

Corollary 4.1 ([24]). *There is a polynomial-time approximation algorithm for the MRS problem on a graph* G *with an approximation ratio of* $\Delta(G)$.

But this is rather unsatisfying. Can we get a little bit better?

4.1. Greedy approaches

The first algorithm, we want to present, is a so-called GREEDY algorithm. This name is motivated by the fact, that such algorithms will always perform as the next step the one promising the most valuable result. The algorithm is based on the results of Theorem 3.6 and 3.7. We call it *Greedy star* for the algorithm adding stars to the solution:

Algorithm Greedy star:
input: a p-edge-coloured graph $G = (V, E)$

output: a rainbow subgraph $H \subseteq G$

S1 choose a vertex v of maximum degree and add all vertices of $N[v]$ to $V(H)$

S2 add the edges between v and its neighbours to $E(H)$ and delete all edges of the same colours in G, continue with step 1 until there are no edges left in G

Theorem 4.1. *The Greedy star algorithm for the MRS problem is* $\left(\frac{\Delta}{2}+\frac{\ln\Delta+1}{2}\right)$*-approximative on graphs with maximum degree* Δ*. Even more, the approximation ratio is given by* $\frac{d}{2}+\frac{\ln\lceil d\rceil+1}{2}$*, if the average degree* d *of a minimum rainbow subgraph is known.*

Proof. Let F be an optimal solution with q vertices and average degree d. Suppose Greedy star has constructed its solution H in t rounds with m_i edges in round i, where $i \in \{1,\dots,t\}$. I.e., in every round i Greedy star added a star $K_{m_i,1}$. Then

$$\frac{|V(H)|}{|V(F)|} \leqslant \frac{\sum_{i=1}^{t}(m_i+1)}{q} = \frac{p+t}{q} = \frac{t}{q}+\frac{d}{2}.$$

We now have to find an upper bound for t.

If we formulate this as a linear programming problem, we have

$$\sum_{i=1}^{\Delta} t_i \to \max$$
$$\text{with } \sum_{i=1}^{k} i\cdot t_i \leqslant k\cdot\frac{q}{2} \text{ for } 1\leqslant k\leqslant\Delta,$$

where t_i denotes for every $i \in \{1,\dots,\Delta\}$ the number of added stars with i edges. Since the maximization being obvious, the side condition arises from the following: If we look at the subgraph of F induced by the edges of the same colours as used as in the stars of size 1 to k added to H, then the maximum degree must not exceed k, for forcing Greedy star to take a larger star instead of at least two smaller ones. But since for every edge in F we need two of the q vertices, we can bound the number of edges by $k\cdot\frac{q}{2}$.

We will now show, that the optimal solution is given by $t_i = \frac{q}{2i}$ for $1 \leqslant i \leqslant \lceil d\rceil$ and $t_{\lceil d\rceil} = \frac{p-(\lceil d\rceil-1)\frac{q}{2}}{\lceil d\rceil}$:

Assume that there are some integers i,j with $1 \leqslant i < j \leqslant \lceil d\rceil$ such that $t_i < \frac{q}{2i}$ and $t_j > 0$. We take the closer look at the minimum j among all values for $j > i$. But this leads to $t_k = 0$ for all $i+1 \leqslant k \leqslant j-1$. Furthermore, we define $\epsilon := \min\left\{\frac{q}{2i}-t_i, \frac{i}{j}t_j\right\}$. Set $t_j' := t_j - \frac{i}{j}\epsilon$ and

$t'_i := t_i + \epsilon$. This feasible solution implies

$$\sum_{k=1}^{\Delta} t_k + (t'_i - t_i) + (t'_j - t_j) = \sum_{k=1}^{\Delta} t_k + \frac{j-i}{j}\epsilon > \sum_{k=1}^{\Delta} t_k.$$

Using the harmonic series

$$H_k = \sum_{i=1}^{k} \frac{1}{i}$$

leads to

$$t \leqslant \frac{q}{2} \cdot H_{\lceil d \rceil} \leqslant \frac{q}{2} \cdot (\ln \lceil d \rceil + 1),$$

and, moreover,

$$\frac{|V(H)|}{|V(F)|} \leqslant \frac{d}{2} + \frac{H_{\lceil d \rceil}}{2} \leqslant \frac{\Delta}{2} + \frac{H_\Delta}{2} \leqslant \frac{\Delta}{2} + \frac{\ln \Delta + 1}{2}.$$

q.e.d.

We will present an example for proving the above result being best possible:

Example 4.1. We construct a family of graphs $\{G_k\}$ for $k \geqslant 2$ as follows: Let G_2 be the disjoint union of the graphs F_2 and I_2, where F_2 is a cycle of order 4 with $V(F_2) = \{v_1, v_2, v_3, v_4\}$, $E(F_2) = \{v_1v_2, v_2v_3, v_3v_4, v_4v_1\}$ and $V(I_2) = \{v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$, $E(I_2) = \{v_5v_6, v_6v_7, v_8v_9, v_{10}v_{11}\}$. For the edge colouring c we choose $c(v_1v_2) = c(v_5v_6) = 1$, $c(v_2v_3) = c(v_8v_9) = 2$, $c(v_3v_4) = c(v_6v_7) = 3$ and $c(v_4v_1) = c(v_{10}v_{11}) = 4$. G_2 contains F_2 as minimum rainbow subgraph, but Greedy star may choose at first the P_3 on the vertices v_5, v_6, v_7 and at last the two single edges v_8v_9 and $v_{10}v_{11}$. That are seven instead of four vertices in the optimal solution.

Given G_{k-1} we construct G_k in the following way: Let G_k be the disjoint union of k copies $G^1_{k-1}, G^2_{k-1}, \ldots, G^k_{k-1}$ of G_{k-1} and $\frac{|V(F_{k-1})|}{2}$ stars $K_{1,k}$. Furthermore we add a matching of $k \cdot \frac{|V(F_{k-1})|}{2}$ edges to the k copies of G_{k-1} in a circular matter: For every F^i_{k-1} divide its vertex set into two halves $U(F^i_{k-1})$ and $W(F^i_{k-1})$ of $\frac{|V(F_{k-1})|}{2}$ vertices each. Now we add the perfect matching between the second half W^i_{k-1} of vertices of G^i_{k-1} and the first half U^{i+1}_{k-1} of G^{i+1}_{k-1} for $i \in \{i, \ldots, k\}$ (considering indices modulo k). Then G_k has maximum degree k.

The subgraph of G_k induced by the k copies of F_{k-1} will be denoted by F_k, the rest of the k copies of G_{k-1} will be I_k.

The edges of each G_{k-1}^i will be coloured with a private set of colours C_{k-1}^i. Next choose a set C_k^* of exactly $k \cdot \frac{|V(F_{k-1})|}{2}$ new colours to colour the edges of the added perfect matching between the k copies of G_{k-1} pairwise distinct. Take the same colour set C_k^* to colour the edges of the $\frac{|V(F_{k-1})|}{2}$ stars $K_{1,k}$ all distinct. In Figure 4.1 the graphs G_2 and G_3 are presented.

Obviously, $V(G[C_{k-1}^i]) \cap V(G[C_{k-1}^j]) = \emptyset$ for $1 \leqslant i < j \leqslant k$. Hence

$$rs(G_k) \leqslant \sum_{i=1}^{k} rs(G_{k-1}^i) = 2 \cdot k \cdot (k-1)! = 2 \cdot k!,$$

for $|V(F_2)| = 4 = 2 \cdot 2!$.

But then F_k is a minimum rainbow subgraph of order $2 \cdot k!$, leading to

$$rs(G_k) = 2 \cdot k!.$$

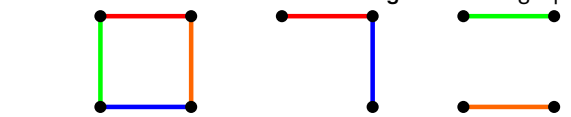
The algorithm Greedy star may first choose the $\frac{|V(F_{k-1})|}{2} = k!$ stars $K_{1,k}$ and then inductively the stars $K_{1,k-1}$ in $G_{k-1}^1, G_{k-1}^2, \ldots, G_{k-1}^k$ and so on. Therefore, the algorithm's worst choice is I_k. Starting with $|V(I_2)| = 7 = 2! \cdot (2 + H_2)$, we get by induction

$$\begin{aligned}
|V(I_{k+1})| &= \frac{|V(F_{k-1})|}{2} \cdot (k+2) + (k+1) \cdot |V(I_k)| \\
&= \frac{2k!}{2} \cdot (k+2) + (k+1) \cdot k! \cdot (k + H_k) \\
&= \frac{k! \cdot (k+1) \cdot (k+2)}{k+1} + \frac{(k+1)!}{k+1}\big((k+1) \cdot (k + H_k)\big) \\
&= \frac{(k+1)!}{k+1} \cdot \big((k+2) + k^2 + k + (k+1) \cdot H_k\big) \\
&= \frac{(k+1)!}{k+1} \cdot \big(k^2 + 2k + 1 + (k+1) \cdot H_k + 1\big) \\
&= \frac{(k+1)!}{k+1} \cdot \big((k+1)^2 + (k+1) \cdot H_k + 1\big) \\
&= (k+1)! \cdot \left((k+1) + H_k + \frac{1}{k+1}\right) \\
&= (k+1)! \cdot \big((k+1) + H_{k+1}\big)
\end{aligned}$$

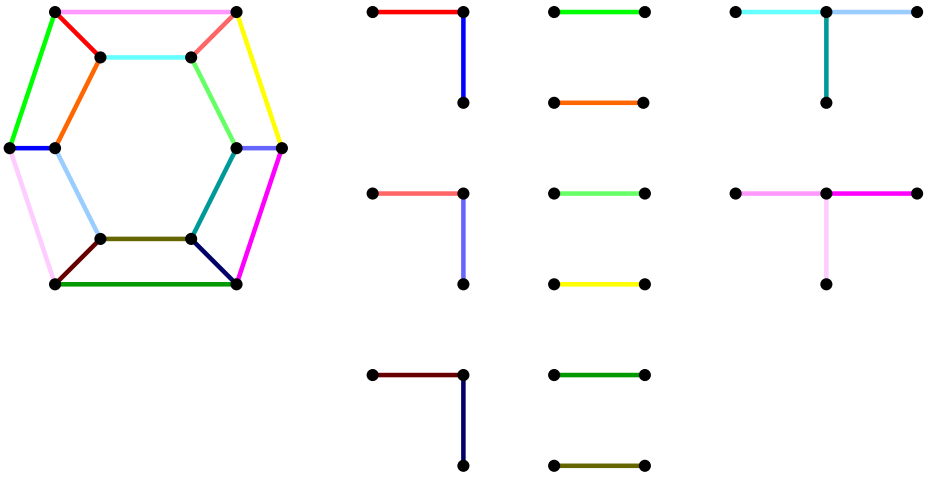
and hence

$$\frac{|V(I_k)|}{|V(F_k)|} = \frac{k}{2} + \frac{H_k}{2}.$$

Fig. 4.1.: The graphs G_2 and G_3



G_2



G_3

Since in the worst case we have to browse through all vertices in G in order to find one of maximum degree, and in the case of a graph consisting only of a matching we have to do this p times, the following corollary concerning the complexity of Greedy star arises easily.

Corollary 4.2. *The Greedy star algorithm has a complexity of pn for p-edge-coloured graphs of order n.*

We will now take a closer look on smaller graphs with respect to their maximum degree:

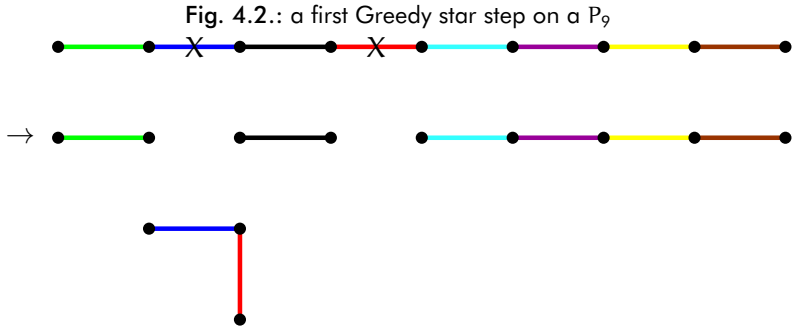
Theorem 4.2. *The Greedy star algorithm is $\frac{7k+2}{4k+2}$-approximative on paths of length $4k+i$, where $i \in \{0,1,2,3\}$. On forests with maximum degree Δ, Greedy star is $\frac{2p+1-\Delta}{p+1}$-approximative. On cycles, the approximation ratio of $\frac{7}{4}$ is sharp.*

Proof. Let p be the number of distinct colours in the graph G.

paths: Assume that the optimal solution is a path of length p. Since in graphs only consisting of paths the maximum degree Δ is 2, Greedy star will always choose a vertex of this degree

as long as there is one. But then the worst case is ending up with the largest possible number of P_2s for the missing last edges. Apparently this happens, if every second edge of the optimal solution is chosen. This leads to a case analysis depending on the number of distinct colours modulo 4. Figure 4.2 illustrates a first Greedy star step on a P_9.

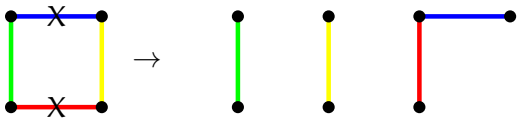
Fig. 4.2.: a first Greedy star step on a P_9



If $p = 4k$, with k being some non-negative integer, then we get k paths of length 2 and $2k$ P_2s. For $p = 4k + 1$ we get again k P_3s, but $2k + 1$ P_2s; in the case of $p = 4k + 2$ exactly $k + 1$ P_3s and $2k$ P_2s. Finally, $p = 4k + 3$ leads to $k + 1$ paths of length 2 and $2k + 1$ ones of length 1. Thus, comparing these four cases, we have an approximation ratio of at most $\frac{7k+2}{4k+2}$, which turns out being slightly better than $\frac{7}{4}$, given by Theorem 4.1.

cycles: If $\delta(G) = \Delta(G) = 2$, then G only consists of cycles. The approximation ratio for these graphs is given by $\frac{7}{4}$. Unfortunately, this is sharp for e.g. a C_4, as shown in Figure 4.3

Fig. 4.3.: Greedy star on a C_4



trees: For trees, the only thing we can guarantee is the choice of a vertex of maximum degree Δ. But then, if the optimal solution is a tree, it could decompose into $\Delta + 1$ new trees. Therefore, we can guarantee a solution of order $2p - \Delta + 1$, leading to an approximation ratio of $\frac{2p+1-\Delta}{p+1}$, which is an approvement with respect to the trivial bound of $\frac{2p}{p+1}$ and the bound given in Theorem 4.1.

q.e.d.

The first algorithm dealing with the MRS problem was presented in [24]. It deals with another well-known graph theoretical substructure: matchings.

Algorithm MRS matching:

input: a p-edge-coloured graph $G = (V, E)$

output: a rainbow subgraph $H \subseteq G$

S1 construct a graph G' with the vertex set $V(G') = \{v_1, v_2, \ldots, v_p\}$ (v_i corresponds to the colour i) and the edges $v_i v_j \in E(G')$ if there exist two adjacent edges $e, f \in E$ with $c(e) = i$ and $c(f) = j$

S2 compute a maximum matching M of order $\beta(G')$ in G'

S3 construct a graph H with $V(H) \subseteq V$ such that each matching edge of M is represented by two adjacent edges in G of the corresponding colours; for each vertex of $V(G')$ not in M take an edge of this colour

Since every edge in G' corresponds to a pair of adjacent edges of distinct colours in G and every colour left out by the matching part of the algorithm is added as a single edge afterwards, the algorithm is proved to be correct. Furthermore, for every matching edge, we take three vertices into our solution. The colours left out are satisfied by at most two new vertices for the single edge, so we have

$$|V(H)| \leqslant 3\beta(G') + 2(p - 2\beta(G')) = 2p - \beta(G').$$

Corollary 4.3. *For the output graph H in the algorithm MRS matching, we have*

$$|V(H)| \leqslant 2p - \beta(G').$$

To study the approximation ratio of MRS matching, we need the following result. The even case was proved by Kotzig [21]. But we present the proof from [24]:

Theorem 4.3. *Let $G = (V, E)$ be a connected graph of size m. Then the edge set E contains exactly $\left\lfloor \frac{m}{2} \right\rfloor$ pairwise edge disjoint paths of order 3.*

Proof. We will prove the result by induction on the size m. If $1 \leqslant m \leqslant 2$, then the statement obviously holds. Now we assume that G has size $m + 1$ for some $m \geqslant 2$. Let $u_1 u_2 = e \in E$ be any edge. Anymore, assume that e is a cut edge; we denote the two subgraphs of $G - e$ by G_1

and G_2, such that $u_1 \in V(G_1)$ and $u_2 \in V(G_2)$. If G_1 or G_2 is of even size, we can partition its edge set into pairwise disjoint paths of order 3. Otherwise both subgraphs are of odd size. Then, the graph $G_1 + e$ has even size and its edge set can be partitioned into pairwise disjoint paths of order 3. Both cases are solved by applying the induction hypothesis on the remaining edges.

Finally assume that $G - e$ stays connected. If e is adjacent to an edge $f \in E$ such that $G - \{e, f\}$ is connected, then the two edges form a path of order 3 and for the remaining graph the statement holds. Hence we may assume that every edge f adjacent to e is a cut edge of $G - e$. Let $f = v_1v_2$ and v_1 be the vertex incident with e. We denote the two subgraphs of $G - \{e, f\}$ by G'_1 and G'_2, such that $v_1 \in V(G'_1)$ and $v_2 \in V(G'_2)$. If one of the subgraphs has even size, the statement holds by induction. If both components G'_1 and G'_2 have odd size, then $G'_1 + e$ and $G'_2 + f$ have both even size and again the statement holds by induction. q.e.d.

For a connected graph G we will denote by $\beta_3(G)$ the number of pairwise edge disjoint paths P_3 contained in $E(G)$. Then the equality $\beta_3(G) = \left\lfloor \frac{m}{2} \right\rfloor$ holds by Theorem 4.3.

Corollary 4.4. *Let $G = (V, E)$ be a graph and $F \subseteq G$ an optimal MRS solution. Then there is some constant $c \in \mathbb{R}$ such that for every solution H derived by the MRS matching algorithm the inequality*

$$\frac{|V(H)|}{|V(F)|} \leqslant 1 + (1 - c)(\Delta - 1)$$

holds, where $\Delta = \Delta(G)$ is the maximum degree of G.

Proof. Assume that F is the disjoint union of subgraphs $F_i \subseteq G$ with $i \in \{1, \ldots, k\}$ for some $k \in \mathbb{N}$ and $|V(F)| = 2p - m'$, $d_i = 2|E(F_i)| - |V(F_i)|$. Then set $\beta_3(F_i) = c_i \cdot d_i$ for some $c_i \in \mathbb{R}$ with $1 \leqslant i \leqslant k$. It follows that $m' = \sum_{i=1}^{k} d_i$ and

$$\begin{aligned}
|V(H)| &\leqslant 2p - \beta(G') \\
&\leqslant 2p - \sum_{i=1}^{k} \beta_3(F_i) \\
&= 2p - \sum_{i=1}^{k} c_i \cdot d_i \\
&\leqslant 2p - \sum_{i=1}^{k} c \cdot d_i \text{ with } c = \min\{c_1, c_2, \ldots, c_k\} \\
&= 2p - c \cdot m'.
\end{aligned}$$

Furthermore $2p - m' = |(V(F)| \leqslant \frac{2p}{\Delta}$ implies $2p - m' \leqslant \frac{m'}{\Delta - 1}$, for

$$\begin{aligned}
|V(F)| \geqslant \frac{2p}{\Delta} &= \frac{|V(F)| + m'}{\Delta} \\
&\rightarrow |V(F)| \cdot \Delta \geqslant |V(F)| + m' \\
&\rightarrow |V(F)|(\Delta - 1) \geqslant m'.
\end{aligned}$$

But this leads to

$$\frac{|V(H)|}{|V(F)|} \leqslant \frac{2p - c \cdot m'}{2p - m'} = 1 + \frac{(1-c)m'}{2p - m'} \leqslant 1 + \frac{(1-c)m'}{\frac{m'}{\Delta - 1}} = 1 + (1-c)(\Delta - 1).$$

q.e.d.

For the case $\Delta = 2$ all possible components in F can only be paths or cycles. So we obtain $c = \frac{1}{3}$ by minimizing over the following:

- if F_i is an even cycle C_{2q} ($q \geqslant 2$), then $c_i = \frac{1}{2}$,
- if F_i is an odd cycle C_{2q+1} ($q \geqslant 1$), then $c_i = \frac{q}{2q+1} \geqslant \frac{1}{3}$,
- if F_i is a path of even order $2q$ ($q \geqslant 2$), then $c_i = \frac{1}{2}$, and
- if F_i is a path of odd order $2q + 1$ ($q \geqslant 1$), then $c_i = \frac{q}{2q-1} > \frac{1}{2}$,

It follows that $\frac{|V(H)|}{|V(F)|} \leqslant \frac{5}{6}\Delta$, being sharp for $\Delta = 2$. But in general the approximation ratio is better:

Theorem 4.4. *The MRS matching algorithm has an approximation ratio of* $\frac{3}{4}\Delta + \frac{1}{2(\Delta+1)}$ *for graphs* $G = (V, E)$ *with maximum degree* $\Delta(G) = \Delta$.

Proof. Let F be a minimum rainbow subgraph. If F is not connected, the following estimation can be applied to all its components for using $f(1) < f(2) < \cdots < f(\Delta)$ with $f(\Delta) := \frac{3}{4}\Delta + \frac{1}{2(\Delta+1)}$.

Assume that F is connected. Then the order of an approximating solution H is maximum, if the size of F is maximum and if all the chosen paths of order 3 and the eventual P_2 are pairwise vertex disjoint. By Theorem 4.3 F can be decomposed into $\frac{q\Delta - i}{4}$ P_3s and $\left\lfloor \frac{i}{2} \right\rfloor$ single edges for $q\Delta \equiv i \mod 4$ with $0 \leqslant i \leqslant 3$. We have the following four cases:

(i) $q\Delta \equiv 0 \mod 4$: Then $|V(H)| \leqslant \frac{3}{4}q\Delta$ implying $\frac{|V(H)|}{|V(F)|} \leqslant \frac{3}{4}\Delta$.

(ii) $q\Delta \equiv 1 \mod 4$: Then $|V(H)| \leqslant 3 \cdot \frac{q\Delta - 1}{4}$ implying $\frac{|V(H)|}{|V(F)|} \leqslant \frac{3}{4}\Delta - \frac{3}{4q}$.

(iii) $q\Delta \equiv 2 \mod 4$: Then $|V(H)| \leqslant 3 \cdot \frac{q\Delta - 2}{4} + 2$ implying $\frac{|V(H)|}{|V(F)|} \leqslant \frac{3}{4}\Delta + \frac{1}{2q}$.

(iv) $q\Delta \equiv 3 \mod 4$: Then $|V(H)| \leqslant 3 \cdot \frac{q\Delta - 3}{4} + 2$ implying $\frac{|V(H)|}{|V(F)|} \leqslant \frac{3}{4}\Delta - \frac{1}{4q}$.

By taking the maximum we obtain $|V(H)| \leqslant \frac{3}{3}\Delta + \frac{1}{2(\Delta+1)}$. q.e.d.

It remains to discuss the complexity of the algorithm:

Corollary 4.5. *The algorithm MRS matching has a complexity of* $\mathcal{O}(np^2 + p^{\frac{5}{2}})$ *for a* p*-edge-coloured graph* G *on* n *vertices.*

Proof. For the matching part in the algorithm we refer to [25], where it is shown that we can get a maximum matching in $\mathcal{O}(\sqrt{|V|} \cdot |E|)$. Since our graph G' has at most $\binom{p}{2}$ edges, we get the $p^{\frac{5}{2}}$ part.

The rest remains for constructing G': for every vertex in G we look at its incident edges. There are at most p of them. Furthermore G is coloured properly, so we have at most $\binom{p}{2}$ pairs of edges resulting in an edge in G'. This gives the np^2 part. q.e.d.

Let us look at graphs, where the MRS consists of a tree. Then MRS matching guarantees a rainbow subgraph H of order

$$|V(H)| \leqslant 2p - \beta(G') \leqslant 2p - \left\lfloor \frac{p}{2} \right\rfloor \leqslant \frac{3}{2}p + \frac{1}{2}.$$

If we soften this assumption and allow a rainbow forest in t components being the optimal solution, then our approximation can be bounded by

$$|V(H)| \leqslant \frac{3}{2}p + \frac{1}{2}t. \qquad (*)$$

Such an approximation will be called ForestApproximation(G) and used in the following greedy algorithm:

Algorithm MRS$_k$ ([17]):
input: a p-edge-coloured graph $G = (V, E)$ and an integer k

output: a rainbow subgraph $H \subseteq G$

S1 find a rainbow subgraph L of order k with at least $k-1$ edges, add L to H and delete all edges of the colours occurring in L – repeat this step, until there is no such subgraph L in G

S2 look for a rainbow cycle L, add L to H and delete all edges of the colours occurring in L – repeat this step until there is no rainbow cycle left

S3 find a ForestApproximation(G) and add it to H

Lemma 4.6 ([17]). *The algorithm MRS$_k$ finds an approximative solution for the MRS problem in a p-edge-coloured graph G on n vertices in $\mathcal{O}(pk^2n^k)$.*

Proof. In step 1 we are interested in rainbow subgraphs of order k with at least $k-1$ edges. Therefore we need to check all k-vertex subsets of V, which are $\binom{n}{k}$. We find at most $\binom{k}{2}$ colours in each of them, so checking for the rainbow means running time of $\mathcal{O}(k^2)$, concluding in $\mathcal{O}(k^2n^k)$ for the first step.

Obviously, the cycles in step 2 have a length of at most $k-1$. We can look for them, if we check all variations of at most $k-1$ vertices, being at most

$$\mathcal{O} = \left(\binom{n}{k-1}(k-1)!\right) = \mathcal{O}(k^2n^2).$$

We repeat the first and this step at most p times, since in every iteration at least one colour is removed.

For the last step we look at Corollary 4.5. This leads to an overall complexity of $\mathcal{O}(pk^2n^2)$.

q.e.d.

For the discussion of the approximation ratio, we will use $(*)$. At first let p_1 denote the number of colours added in step 1 and step 2. Then $p_3 := p - p_1$. Furthermore, let t_3 denote the number of components in $F \cap G$ during step 3, where F is an optimal solution for G. Assume that $k \geqslant 2$. Then the number of vertices during step 1 and step 2 is at most $\left(1 + \frac{1}{k-1}\right) p_1$, and the number of vertices added during step 3 is bounded from above by $\frac{3}{2}p_3 + \frac{1}{2}t_3$. This gives for the approximation H:

$$|V(H)| \leqslant \left(1 + \frac{1}{k-1}\right) p_1 + \frac{3}{2}p_3 + \frac{1}{2}t_3. \qquad (**)$$

Since $F \cap G$ is a forest in step 3, we can bound p_3 by $p_3 \leqslant |V(F)| - 1$. Anymore, $\left(1 + \frac{1}{k-1}\right) \leqslant \frac{3}{2}$, for $k \geqslant 3$, leads to the right side of $(**)$ being maximal for $p_3 = |V(F)| - t_3$ and $p_1 = p - |V(F)| + t_3$. We obtain

$$\begin{aligned} |V(H)| &\leqslant \left(\frac{k}{k-1}\right)(p - |V(F)| + t_3) + \frac{3}{2}(|V(F)| - t_3) + \frac{1}{2}t_3 \\ &= |V(F)|\left(\left(\frac{k}{k-1}\right)\frac{p}{|V(F)|} - 1 - \frac{1}{k-1} + \frac{3}{2}\right) + \left(\frac{1}{k-1}\right)t_3 \\ &\leqslant |V(F)|\left(\left(\frac{k}{k-1}\right)\frac{p}{|V(F)|} + \frac{1}{2}\right). \end{aligned}$$

With this calculation, we can prove the following theorem:

Theorem 4.5 ([17]). *If G is a p-edge-coloured graph and F is an optimal solution to the MRS problem on G, then MRS_k computes a solution H having at most $\frac{1}{2}|V(F)| + \left(1 + \frac{1}{k-1}\right) p$ vertices.*

The integer k leaves us the possibility to tune the approximation ratio of the rainbow solution for a given graph.

Corollary 4.7 ([17]). *For every positive integer $\varepsilon > 0$ there is a polynomial time approximation algorithm for the MRS problem, such that the approximation ratio can be bounded from above by*

$$\frac{1}{2} + \left(\frac{1}{2} + \varepsilon\right)\Delta,$$

for graphs with maximum degree Δ.

Proof. Assume F being an optimal solution on p edges. For every vertex in F the degree is at most Δ. Therefore, $\frac{p}{|V(F)|} \leqslant \frac{\Delta}{2}$, according to the Handshaking Lemma. Using Theorem 4.5 we get an approximation ratio of

$$\frac{1}{2} + \left(1 + \frac{1}{k-1}\right)\frac{\Delta}{2} \leqslant \frac{1}{2}\left(\frac{1}{2} + \frac{1}{k}\right)\Delta.$$

Now it only remains to choose k sufficiently large in order to get $\frac{1}{k} \leqslant \varepsilon$. q.e.d.

4.2. The local colour density

Until now we used rather pragmatic approaches. If the algorithm had to choose between several vertices, we selected the candidate by taking the first one given out of some maybe god-given order. We are interested in making this choice somehow more comprehensive in order to reduce the complexity and improve the run-time.

Let us take a closer look at the neighbourhood of the chooseable vertices:

Definition 4.1. Let $G = (V, E)$ be a graph and $c : E \to \{1, 2, \ldots, p\}$ a p-edge-colouring of G. Then the *local colour density* $\mathrm{lcd}(S, T)$ of two subsets $S, T \subseteq V$, $S \cap T = \emptyset$ is defined in the following way

$$\mathrm{lcd}(S, T) = \frac{|c(G[S \cup N[T]])| - |c(G[S])|}{|N[T] - S|}.$$

Obviously, the local colour density is a measure on the proportion of how many colours will be added if we take a given set of vertices. If $T = \{v\}$ is a single vertex, $S = \emptyset$ and c a proper edge

colouring, then we have the simple bounds

$$\frac{\delta}{\delta+1} \leqslant \mathrm{lcd}(v) \leqslant \frac{n-1}{2},$$

where $\mathrm{lcd}(v) := \mathrm{lcd}(\emptyset, \{v\})$. In the same way we can define the local colour density of a single edge $e = uv$ with $T = \{u, v\}$ and $S = \emptyset$, denoting it by $\mathrm{lcd}(e) := \mathrm{lcd}(\emptyset, e)$. Following this gives

$$\frac{\max\{d(u), d(v)\}}{n} \leqslant \mathrm{lcd}(e) \leqslant \frac{n-1}{2}.$$

A very simple approach for using the local colour density as argument of choice is the following one:

Algorithm SortLCD ([12]):
input: a p-edge-coloured graph $G = (V, E)$

output: a rainbow subgraph $H \subseteq G$

S1 for every edge $e \in E$ calculate $\mathrm{lcd}(e)$ and sort the edges by decreasing local colour density and separately by colour

S2 choose an edge of maximum lcd for H

S3 for every colour choose an edge in the neighbourhood of H, such that the local colour density is maximum; if there is no such edge, take an edge of maximum lcd

Concerning the running time, the most expensive part of the algorithm is step 1. The graph has at most $\binom{n}{2}$ edges, where every end vertex has at most p neighbours, so we get an ordering of the lcd values in $\mathcal{O}(p^2n^2)$. Step 3 is mainly checking the neighbourhood of the ordered edges, which can be done for every one of the p colours in $\mathcal{O}(p)$.

Corollary 4.8. *In a p-edge-coloured graph of order n the algorithm SortLCD has a complexity of $\mathcal{O}(p^2n^2)$.*

It turns out, that calculating the local colour density once is not very efficient, since the tenor of introducing it was to assure, that the next added edge will provide the largest number of new colours by adding as few as possible new vertices. We only used empty sets for S instead of

introducing the already added vertices. Therefore we will not discuss any approximation results, but instead of we introduce the following algorithm paying more attention to the definition of the local colour density:

Algorithm LCD_k:

input: a p-edge-coloured graph $G = (V, E)$ and an integer k

output: a rainbow subgraph $H \subseteq G$

S1 find a subset $T \subseteq V$ with $1 \leqslant |T| \leqslant k$, such that the value $lcd(H, T)$ is maximum

S2 add the graph induced by $N[T]$ to H, continue until H contains exactly one edge of each colour

Obtaining the approximation ratio for LCD_k is a much more complicated task than for MRS matching or Greedy star. We will only give an upper bound:

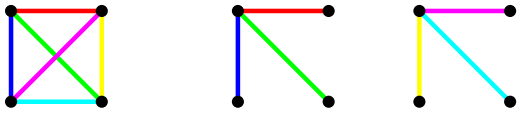
Theorem 4.6. *The algorithm LCD_k finds a $\left(\frac{\Delta}{2} + \frac{\ln \Delta + 1}{2}\right)$-approximative solution for graphs with maximum degree Δ. If the average degree d of an optimal solution is known, then the approximation ratio is $\frac{d}{2} + \frac{\ln \lceil d \rceil + 1}{2}$.*

Proof. Basically we can apply the proof of Theorem 4.1 for the algorithm Greedy star. The worst case occurs if the edges added in one step by LCD_k induce a tree. So they may induce a star and we follow the proof of Theorem 4.1. q.e.d.

Note that if the graph is triangle-free, then LCD_1 behaves in almost the same way as Greedy star would do. Therefore, we can give the same graphs as proof for sharpness as given in Example 4.1. On the other hand, LCD_1 can indeed lead to a minimum rainbow subgraph, where for example Greedy star fails. Look at the following graph $G = (V, E)$:

Example 4.2. The order of G is $n = 12$ with the vertices $V = \{v_1, v_2, \ldots, v_{12}\}$ and the edge set $E = \{v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4, v_3v_4, v_5v_6, v_5v_7, v_8v_8, v_9v_{10}, v_9v_{11}, v_9v_{12}\}$. For the edge colouring we choose $c(v_1v_2) = c(v_5v_6) = 1$, $c(v_1v_3) = c(v_5v_7) = 2$, $c(v_1v_4) = c(v_5v_8) = 3$, $c(v_2v_3) = c(v_9v_{10}) = 4$, $c(v_2v_4) = c(v_9v_{11}) = 5$ and $c(v_3v_4) = c(v_9v_{12}) = 6$. The graph is illustrated in Figure 4.4.

Fig. 4.4.: LCD_1 can work optimal



This example can be generalized to a class of graphs $\mathcal{G}$, where LCD_1 always finds a minimum rainbow subgraph: every graph $G \in \mathcal{G}$ contains a minimum rainbow subgraph of order $k = \Delta + 1$, which is complete or missing less than q edges. To determine q we are looking at the upper bound of the lcd value of a single vertex $v \in V$, which is given by

$$lcd(v) \leqslant \frac{|N[v]| - 1}{2}.$$

In order to force the algorithm to choose one of the vertices in the minimum rainbow subgraph, the following must hold:

$$\frac{\frac{k(k-1)}{2} - q}{k} > \frac{k-2}{2} \rightarrow \frac{k^2 - k - 2q}{2k} > \frac{k-2}{2} \rightarrow q < \frac{k}{2}.$$

At last it remains to take a closer look on the complexity of the algorithm. Checking all subsets on k vertices can be done in $\mathcal{O}(n^k)$ running time. For every of these subgraphs we have to calculate the local colour density. This can be done in $\mathcal{O}(p^2k^2)$ for being $\mathcal{O}(kp)$ vertices in the closed neighbourhood of this subgraph. Since in every step at least one edge is added, we have at most p steps. This gives a proof to the following corollary:

Corollary 4.9. *The algorithm LCD_k has a complexity of $\mathcal{O}(p^3k^2n^k)$ for a p-edge-coloured graph G on n vertices.*

In [12] some other algorithms using the local colour density are presented. They were developed out of SortLCD, but are still greedy approaches. So-called zones are introduced. Depending on how close an edge lies to the overlooked neighbourhood (if both end vertices are in the neighbourhood or just one of them) the next added edge is chosen. Furthermore a recalculating of the lcd values is considered, leading to a much larger complexity, but no significant better approximation ratio. Therefore we renounce these algorithms.

4.3. MaxNewColour

In this section we discuss a various number of algorithms, developed during the internship of Franziska Heinicke ([12]). The goal was to provide some "small" algorithms. They should run very fast in order to being used quite a lot of times for the same problem. Therefore, in most of them some random element was implemented and in the end the best solution was issued after some passes.

We start with the basic algorithm:

Algorithm MaxNewColour ([12]):
input: a p-edge-coloured graph $G = (V, E)$

output: a rainbow subgraph $H \subseteq G$

S1 choose a vertex $v \in N(H)$, such that the degree $d(v, G[H \cup \{v\}])$ is maximal; add v and the edges between v and H to H and delete all edges of these colours in G and the vertex v

S2 if there is no v in the neighbourhood of H left, choose an edge $e \in E$ to start a new component

We slightly altered step 1 as opposed to [12]. Now the algorithm deletes every chosen colour and vertices for performance reasons.

In step 2 we have the possibility of different arguments of choice. We will also use this step to start the algorithm, for H being empty at the beginning and therefore the neighbourhood of H being empty, too. The most simple argument is to choose the next edge randomly. But this approach seems to be very poor. As we have seen before, a well chosen starting edge can accelerate the algorithm and rises the chance for a better approximation ratio. On the other hand, this argument of choice is very cheap in terms of running time. Therefore, we execute the algorithm on the same problem for some number of passes and take the smallest solution. When we refer to MaxNewColour, we will always talk about the random choice version.

For complexity considerations we only need to take a closer look at step 1. The first edge is chosen in step 2, then for every vertex in H, which are at first 2, then 3, 4, ... vertices, the neighbourhood is examined. From time to time, when there are no usable neighbours left, we

add two instead of one new vertex in step 2. Thus we have to check $\mathcal{O}(\Delta p^2)$ times, leading to an overall complexity of $\mathcal{O}(p^3)$, since $\Delta \leqslant p$.

Another reason for a simple argument of choice like randomness arises from experimental data: In [12] it was tested how often step 2 was actually used. Given random graphs on n vertices with an edge probability p, we used two different approaches to colour the edges properly. KF1 was a simple random colouring with $2 \cdot \Delta(G) - 1$ colours. Here every colour appears almost equally. KF2 is a Greedy approach. It will start with edges with a large neighbourhood and uses the smallest available colour. Obviously, this colouring results in a much smaller number of used colours. Table 4.1 provides these results.

Tab. 4.1.: frequency of using step 2 in "smooth" graphs

	$n = 10$ $p = 0.2$	$n = 10$ $p = 0.5$	$n = 25$ $p = 0.2$	$n = 25$ $p = 0.5$	$n = 50$ $p = 0.2$	$n = 50$ $p = 0.5$	$n = 50$ $p = 0.8$
KF1	1.38	1.16	1.34	1.12	1.12	1	1
KF2	1.12	1.04	1.32	1.22	1.28	1.3	1.1

Remark. We call these used graphs "*smooth*" for the edges being uniformly distributed with probability p. In addition to this we define "*non-smooth*" graphs, which arise from k random ("smooth") graphs components connected by at most six edges.

Even for "non-smooth" graphs, the average number using step 2 turns out to be small. Table 4.2 shows the results with such graphs on n vertices.

Tab. 4.2.: frequency of using step 2 in "non-smooth" graphs

	$n = 15$ $k = 2$	$n = 15$ $k = 3$	$n = 25$ $k = 3$	$n = 25$ $k = 4$	$n = 25$ $k = 5$
KF1	1.34	1.41	1.3	1.48	1.45
KF2	1.23	1.14	1.13	1.22	1.31

The number of vertices in the graphs is small due to the fact, that the algorithms were implemented in MAPLE. Nonetheless, it is noticeably that the algorithm has to start a new component very rarely.

Another simple argument of choice is realized in the version we call **MaxNewColourDelta**. The goal is to choose an edge with a large neighbourhood. Therefore, at the beginning we calculate the size of every edge's neighbourhood and sort the edges according to that. If H turns out not being expandable in step 1, we choose an edge of an unused colour, having the largest neighbourhood at the beginning of the algorithm. In this case the complexity differs only at the beginning. Building up the edge ordering consists mainly of sorting, which can be done in $\mathcal{O}(np \cdot \ln np)$. Thus the overall complexity results in $\mathcal{O}(p^3 + np \cdot \ln np)$.

Up to now the versions of the algorithm did not pay any attention to the number of colours that a new vertex provides in the next step. Now we will change this. In the version **MaxNewColourBunt** the algorithm will take, out of the set of vertices adding the largest number of new colours to H, that one with the largest neighbourhood. In step 2 we choose an edge which the largest number of adjacent new colours. Therefore we not only need to check the degree of a vertex v with respect to H, but also outside $H \cup \{v\}$. But this can be done in the same step, so we end up again with $\mathcal{O}(np \cdot \ln np)$.

For the last variant we thought of starting with colours, that appear only a few times. Frequent colours are simple being added, because the chance of finding them in the neighbourhood of H is large. So we start with counting the number of appearance for each colour and sorting this list. Every time the algorithm is forced to execute step 2, it will take a random edge of a colour, which is not included in H until now, but occurs as rare as possible. This algorithm was designed, likewise the basic one, to be executed several times. Here the big advantage is, that the sorting step is only needed once. This variant is called **MaxNewColourFH**.

The complexity differs from the one of the basic algorithm in counting and sorting, which can be realized in $\mathcal{O}(np + p \log \ln p)$. This leads to $\mathcal{O}(np + p^3)$.

We will merge these results in the following theorem:

Theorem 4.7. *Let G be a proper p-edge-coloured graph on n vertices and with maximum degree Δ. Then the variants of the algorithm MaxNewColour have the run-time complexities as shown as in Table 4.3.*

Remark. We did not pay any attention to the part, where the algorithm deletes the already added colours. We state that the complexity concerning this step can be neglected by using effective

Tab. 4.3.: run time complexity for MaxNewColour and its variants

MaxNewColour	$\mathcal{O}(p^3)$
MaxNewColourDelta	$\mathcal{O}(p^3 + np \cdot \ln np)$
MaxNewColourFH	$\mathcal{O}(np + p^3)$
MaxNewColourBunt	$\mathcal{O}(p^3 + np \cdot \ln np)$

data structures handling the graph. Otherwise deleting would mean checking every edge in G for its colour, which can be done in $\mathcal{O}(np)$.

Compared with the complexities in [12], the results presented vary. The reason is that we tried to make a much closer estimation and constrained ourselves to results only using the number of different colours p and the maximum degree Δ as parameters. Surely, the maximum degree could be approximated by p as well, but in our opinion this would lead to a way too loose approximation. Furthermore, we did not consider special implementations using MAPLE, as done in the paper.

In [12] the variants of MaxNewColour were compared with respect to their running time on some example graphs. The algorithms were tested on the two different colourings KF1 and KF2 given above. In order to compare the algorithms, a set of graphs was created and all algorithms were executed on these ones. Afterwards we calculated the average order of the solutions found. If there are numbers in the brackets behind an algorithm, then it means, that we have run this one as many times as given. The results are presented in the following four tables 4.4 to 4.7.

Let's have a closer look at the approximation ratio of the several versions of the algorithm. For MaxNewColour we are not able to guarantee a better solution than the worst case solution, namely p single edges, one for each colour. Since the basic version chooses the starting edge and the ones in step 2 randomly, we can say that MaxNewColour has a guaranteed approximation ratio of $\sqrt{2p}$, using the lower bound of Fact 2.2. The same is true for MaxNewColourFH, since the edge in step 2 is, again, chosen randomly in the set of edges of a fewest appearing colour.

In the case of MaxNewColourDelta we choose an edge with largest neighbourhood in step 2.

Tab. 4.4.: average order of the solutions with MaxNewColour in "smooth" graphs coloured by KF1

	$n = 25$ $p = 0.2$	$n = 25$ $p = 0.5$	$n = 50$ $p = 0.2$	$n = 50$ $p = 0.5$	$n = 50$ $p = 0.8$
MaxNewColour	12.6	13.9	17.3	19.4	20.2
MaxNewColourDelta	12.2	13.7	16.9	19.2	20.3
MaxNewColourFH	12.0	13.5	17.0	19.4	20.1
MaxNewColourBunt	11.6	13.3	16.3	18.7	19.6
MaxNewColour (20)	11.2	12.6	15.5	18.0	19.0
MaxNewColour (50)	11.1	12.4	15.3	17.8	18.8
MaxNewColourFH (20)	11.6	12.9	15.9	18.2	19.1

Tab. 4.5.: average order of the solutions with MaxNewColour in "smooth" graphs coloured by KF2

	$n = 25$ $p = 0.2$	$n = 25$ $p = 0.5$	$n = 50$ $p = 0.2$	$n = 50$ $p = 0.5$	$n = 50$ $p = 0.8$
MaxNewColour	8.2	9.6	11.8	14.4	16.5
MaxNewColourDelta	7.5	9.7	10.9	14.1	16.8
MaxNewColourFH	7.4	8.7	10.5	13.1	15.7
MaxNewColourBunt	7.3	8.8	10.3	13.0	15.6
MaxNewColour (20)	6.9	8.3	10.0	12.6	14.9
MaxNewColour (50)	6.8	8.2	9.8	12.4	14.7
MaxNewColourFH (20)	7.1	8.5	10.2	12.8	15.4

Remember that no further attention is paid on the number of distinct colours adjacent to this edge. Therefore the worst case appears if a vertex of this edge is incident to the same colours as the other one. This looks familiar. It is almost the same situation as choosing a star of largest cardinality, leading to the approximation ratio of Theorem 4.1.

Now consider MaxNewColourBunt. If we need step 2, then an edge with the largest number of unused colours adjacent to it is chosen. Assume that the minimum rainbow subgraph is a complete graph. Then at the beginning of the algorithm the neighbourhood of every edge

Tab. 4.6.: average order of the solution with MaxNewColour in "non-smooth" graphs coloured by KF1

	$n = 25$ $k = 3$	$n = 25$ $k = 4$	$n = 25$ $k = 5$
MaxNewColour	11.5	11.4	11.6
MaxNewColourDelta	11.3	11.4	11.7
MaxNewColourFH	10.8	10.9	11.2
MaxNewColourBunt	10.4	10.3	10.6
MaxNewColour (20)	10.0	10.0	10.2
MaxNewColour (50)	10.0	9.9	10.2
MaxNewColourFH (20)	10.3	10.4	10.7

Tab. 4.7.: average order of the solution with MaxNewColour in "non-smooth" graphs coloured by KF2

	$n = 25$ $k = 3$	$n = 25$ $k = 4$	$n = 25$ $k = 5$
MaxNewColour	7.92	8.02	7.94
MaxNewColourDelta	7.78	7.92	7.79
MaxNewColourFH	6.96	7.04	6.85
MaxNewColourBunt	6.95	6.96	6.86
MaxNewColour (20)	6.56	6.64	6.48
MaxNewColour (50)	6.50	6.58	6.43
MaxNewColourFH (20)	6.83	6.83	6.67

consists of exactly $2(n-2)$ unused colours. The worst case arises if the algorithm chooses an edge e with this configuration, but no other colour in $N[N[e]]$. Therefore it added $2 + 2(n-2)$ vertices and $2(n-2)+1$ colours to H and

$$\binom{n}{2} - (2n-3) = \frac{n^2 - 5n + 6}{2} = \binom{n-2}{2}$$

edges, respectively colours, are left. A new edge is needed and we can apply this procedure

again, assuming that the optimal solution for the rest is complete. It leads to

$$|V(H)| \leqslant \sum_{k=1}^{\lceil \frac{n-1}{2} \rceil} 2 + 2(n-2k) = \begin{cases} \sum_{k=1}^{\frac{n}{2}} 4k - 2 = \frac{n^2-4n}{2} & \text{, if } n \text{ even,} \\ \sum_{k=1}^{\frac{n-1}{2}} 4k = \frac{n^2-3n+2}{2} & \text{, if } n \text{ odd.} \end{cases}$$

For this reason the approximation ratio is $\frac{\Delta}{2} + \frac{\Delta}{2(\Delta+1)}$.

The next theorem provides an overview over the complexities:

Theorem 4.8. *Let* G *be an* p*-edge-coloured graph with maximum degree* $\Delta(G) = \Delta$ *and an optimal MRS-solution* F*. Then the variants of MaxNewColour provide solutions with the following approximation ratios:*

Tab. 4.8.: approximation ratios of MaxNewColour

MaxNewColour	$\sqrt{2p}$
MaxNewColourDelta	$\frac{\Delta}{2} + \frac{\ln \Delta + 1}{2}$
MaxNewColourFH	$\sqrt{2p}$
MaxNewColourBunt	$\frac{\Delta}{2} + \frac{\Delta}{2(\Delta+1)}$

5. What is real data telling us?

As seen before, the foundations of the MRS problem came from the bioinformatics. Therefore, instead of studying MRS on general graphs, graphs derived from real data, i.e., graphs constructed out of a given set of genotypes, may have some certain properties leading to more efficient heuristics attacking MRS. For example, instances only consisting of genotypes with at most one SNP would lead to graphs, where every colour appears exactly once.

In the following chapter we will discuss structural characteristics of graphs corresponding to a set of genotypes. For short, we call such graphs *real data graphs*. Notwithstanding, these graphs will not necessarily refer to real biological problem instances of pure parsimony haplotyping.

We start with a closer look at the number of 2-entries in the genotypes:

Theorem 5.1. *Let* $G = (V, E)$ *be a real data graph and* $\mathcal{G}$ *the corresponding set of genotypes. Then* G *is bipartite if there is no genotype* $g \in \mathcal{G}$ *with an even number of* 2*-entries.*

Proof. For the proof we will identify the vertices of G with the corresponding haplotypes explaining the genotypes of $\mathcal{G}$. Suppose that all genotypes $g \in \mathcal{G}$ have an even number of 2-entries.

At first assume there is a triangle consisting of the three vertices $a, b, c \in V$. W.l.o.g. let a and b be of the form ($i, j, k, l, m \in \mathbb{N}$)

$$a = \overbrace{\underbrace{11\ldots11}_{2j+1}\underbrace{11\ldots11}_{2k}}^{2i+1}\underbrace{00\ldots00}_{2l+1}\underbrace{00\ldots00}_{2m}00\ldots$$

$$b = \overbrace{\underbrace{00\ldots00}_{2j+1}\underbrace{00\ldots00}_{2k}}^{2i+1}\underbrace{00\ldots00}_{2l+1}\underbrace{00\ldots00}_{2m}00\ldots$$

The number at the braces indicates the length of this block in the vector. Let $|x|_r$ denote the number of rs in the entries of the vector x.

In order to get an odd number of 2s in $b + c$, there are the following possible cases:

(i)

$$c = \overbrace{\underbrace{00\ldots00}_{2j+1}\underbrace{00\ldots00}_{2k}}^{2i+1}\underbrace{11\ldots11}_{2l+1}\underbrace{00\ldots00}_{2m}00\ldots$$

But then $|(a + c)|_2 = (2i + 1) + (2l + 1)$ is even, a contradiction.

(ii)

$$c = \overbrace{\underbrace{00\ldots00}_{2j+1}\underbrace{11\ldots11}_{2k}}^{2i+1}\underbrace{11\ldots11}_{2l+1}\underbrace{00\ldots00}_{2m}00\ldots$$

But then, again, $|(a + c)|_2 = (2j + 1) + (2l + 1)$ is even.

(iii)

$$c = \overbrace{\underbrace{11\ldots11}_{2j+1}\underbrace{00\ldots00}_{2k}}^{2i+1}\underbrace{00\ldots00}_{2l+1}\underbrace{11\ldots11}_{2m}00\ldots$$

This time $|(a + c)|_2 = 2k + 2m$, which is, of course, an even number.

Therefore such a triangle does not exist.

By the way, in almost the same way we can conclude that there is no triangle consisting of two genotypes with an even number of 2s and only one with an odd, say $|a+b|_2$ and $|a+c|_2$ is even, but $|b + c|_2$ is odd. Let here w.l.o.g. a and b of the form ($i, j, k, l, m, n \in \mathbb{N}$):

$$a = \overbrace{\underbrace{11\ldots11}_{2j+1}\underbrace{11\ldots11}_{2k+1}\underbrace{11\ldots11}_{2l}}^{2i}\underbrace{00\ldots00}_{2m+1}\underbrace{00\ldots00}_{2n}00\ldots$$

$$b = \overbrace{\underbrace{00\ldots00}_{2j+1}\underbrace{00\ldots00}_{2k+1}\underbrace{00\ldots00}_{2l}}^{2i}\underbrace{00\ldots00}_{2m+1}\underbrace{00\ldots00}_{2n}00\ldots$$

(i)

$$c = \overbrace{\underbrace{11\ldots11}_{2j+1}\underbrace{00\ldots00}_{2k+1}\underbrace{00\ldots00}_{2l}}^{2i}\underbrace{11\ldots11}_{2m+1}\underbrace{00\ldots00}_{2n}00\ldots$$

But then $|b + c|_2 = (2j + 1) + (2m + 1)$ is even, too.

(ii)

$$c = \overbrace{\underbrace{00\ldots00}_{2j+1}\underbrace{00\ldots00}_{2k+1}\underbrace{11\ldots11}_{2l}}^{2i}\underbrace{00\ldots00}_{2m+1}\underbrace{11\ldots11}_{2n}00\ldots$$

Here $|b+c|_2 = 2l+2n$ is even, again.

But a triangle with exactly two genotypes having an odd number of 2s is possible, see

$$a = 111,\ b = 000,\ c = 100.$$

Furthermore, a triangle consisting of genotypes with an even number of 2-entries is possible, too. Here you may consider

$$a = 000,\ b = 101,\ c = 110.$$

Now we take a closer look on an arbitrary odd cycle $C_k = v_1v_2\ldots v_kv_1$. Then only an even number of genotypes $g_i = v_i \oplus v_{i+1}$, where $i \in \{1,2,\ldots,k\}$ mod k, may have a 2 at site j. Otherwise, consider this number t being odd. If $t = k$ and site j is 0 in v_1, then this site has to be a 1 in every haplotype of even order and a 0 in every haplotype of odd order. Following this we must have a 1 in v_1, a contradiction.

Thus, if t is odd, it has to be less than k. But then we can decompose the C_k into paths $P_1, P_2, \ldots P_s$, for some $s \in \mathbb{N}$, such that $P_1 = v_1v_2\ldots v_a$, $P_2 = v_av_{a+1}\ldots v_b$, $\ldots$, $P_s = v_cv_{c+1}\ldots v_kv_1$, with $1 < a < b < c \leqslant k$, and the corresponding genotypes alongside one path equal at position j. Furthermore, the ambiguous position j differs in paths sharing an end vertex.

Obviously, there has to be at least one path of odd length with the genotypes having a 2 at position j. Assume P_1 is such a path. Then the end vertices of P_1 differ at the jth position, say for v_1 we have a 0, respectively for v_a we have a 1. If the rest of the paths with a 2 in the genotypes at site j are of even length, then this site is a 1 in P_s, a contradiction. But if there is another path of odd length with a 2 at the ambiguous site, then we have an even number of genotypes with a 2-entry at position j and we need a third path of odd length and so on.

We see, for each position j we have an even number of genotypes with a 2 at this ambiguous site in our C_k. Thus, if we have $2l$ genotypes $\mathcal{G}^* = \{g_1, g_2, \ldots, g_{2l}\}$ (l being an integer less than $\frac{k}{2}$)

with a 2-entry at position one, then we may have one of the following three cases for the second site:

- None $g \in \mathcal{G}^*$ has a 2 at the second position: Then we have $2l'$ other genotypes having a 2 at this position, where l' is an integer less than $\frac{k}{2} - l$. But now we end up with an even number of genotypes with a 2 at one of the first two sites, which is similar to the situation at the begin of the case analysis.
- An even number of genotypes $g \in \mathcal{G}^*$ has a 2-entry at the second position. Then, only considering the first and second site, we have an even number of genotypes with two 2-entries, and an even number with one 2-entry, which is again the situation from the start.
- An odd number of genotypes $g \in \mathcal{G}^*$ has a 2-entry at the second postion. But then there has to be an odd number of genotypes with a 2 at the second, but no 2 at the first position. This again ends up in an even number of genotypes with exactly one 2-entry in the first two positions.

We can generalize this procedure and will always end up with an even number of genotypes having an odd number of 2-entries, which is a contradiction to the existence of such a C_k. Therefore G has to bipartite. q.e.d.

Another interesting question is the distribution of the colours in G. If you guarantee, e.g., that a colour is not likely to be repeated in the closer neighbourhood of a vertex, then taking larger subgraphs in the heuristics would not impair the approximation ratio that much.

Theorem 5.2. *Let $G = (V, E)$ be a real data graph and $\mathcal{G}$ the corresponding sets of genotypes. Then there are no dichromatic cycles or paths of length 4 or longer.*

Proof. First assume that there is a dichromatic $C_4 = abcda$ with $a, b, c, d \in V$. Let us denote by x_i the ith position of the corresponding haplotype to the vertex $x \in \{a, b, c, d\}$. W.l.o.g. assume that $a_i + b_i = c_i + d_i$ and $b_i + c_i = d_i + a_i$, since ab, cd and bc, da are coloured with the same colour. Let i be fixed for the following enquiry:

(A) $b_i + c_i = 2$

(i) If $a_i + b_i = 2$, too, then also $c_i + d_i = 2$.This leads to $a_i = c_i$ and $b_i = d_i$.

(ii) For $a_i + b_i \neq 2$, we have $a_i = b_i$ and of course $c_i = d_i$. With $a_i + b_i = c_i + d_i$, this is a contradiction to $b_i \neq c_i$.

(B) $b_i + c_i \neq 2$, i.e., $b_i = c_i$.

(i) If $a_i + b_i = 2$, then $a_i \neq b_i$ and $c_i \neq d_i$, leading to $b_i + c_i \neq a_i + d_i$. I.e., there has to be a third colour, a contradiction.

(ii) On the other hand, if $a_i + b_i \neq 2$, then $a_i = b_i = c_i = d_i$.

Only a combination of the cases A (i) and B (ii) could lead to a dichromatic C_4, but as you see, this would mean that $a_i = c_i$ and $b_i = d_i$, for all positions i in the corresponding haplotypes, a contradiction.

Now consider a dichromatic $P_5 = abcde$ with $a, b, c, d, e \in V$ and $a_i + b_i = c_i + d_i$, $b_i + c_i = d_i + e_i$. As seen before, if $b_i + c_i = 2$ and ab, cd are of the same colour, i.e., the corresponding haplotypes can form the same genotype, then $a_i + b_i$ and $c_i + d_i$ must be a 2, too. The same is true for $c_i + d_i = 2$, meaning if we have a 2-entry in any of the colours, then there is a 2 at the same position in the other colour, too. Furthermore, remembering case B (ii), the same appears for non-2-entries. Therefore, if there is a dichromatic P_5, then the five vertices cannot be distinct. q.e.d.

Corollary 5.1. *Let G be a real data graph and $G' \subset G$ a subgraph of G. If $G' \cong K_4$, then $rs(G') \geqslant 5$.*

The question remains, what happens for larger complete subgraphs. For small instances of n, you can simply try and count the number of colours in a K_n. If we denote the minimum number of colours, that you find in a complete subgraph K_n of a real data graph G, by $rd(G)$, then in Table 5.1 this number is given for complete real data graphs on two up to seven vertices.

Tab. 5.1.: minimum number of distinct colours in complete real data graphs

$rd(K_2) = 1$	$rd(K_5) = 9$
$rd(K_3) = 3$	$rd(K_6) = 12$
$rd(K_4) = 5$	$rd(K_7) = 16$

Certainly it would be more convenient to find a closed formula for this number instead of ruling it out by try and error. The most promising subgraphs for such a formula seem to be subgraphs, where the order is a power of 2. It appears that in this case $\mathrm{rd}(K_{2^n}) \geqslant 3^n - 2^n$. The conjecture arises from genotypes of length n. If the set $\mathcal{G}$ consists of all possible genotypes of length n, then it corresponds to the complete graph of order 2^n with an edge colouring on $3^n - 2^n$ colours.

Conjecture 1. Let $G = (V, E)$ be a real data graph and $K \subset G$ a complete subgraph of G. Then the minimum number of colours in K is given by

$$\mathrm{rd}(K) \geqslant \begin{cases} 3^n - 2^n & \text{, if } n(K) = 2^n \\ 3^n - 2^n - n & \text{, if } n(K) = 2^n - 1 \\ 3^n & \text{, if } n(K) = 2^n + 1. \end{cases}$$

For a short motivation of these bounds, consider in the case $n(K) = 2^n - 1$ that we delete a vertex of a real data K_{2^n}. Then the best chance to end up with less colours is that we delete in this procedure a lot of edges of single occurring colours. Since the corresponding haplotype of this deleted vertex is fixed and a single colour refers to a genotype with exactly one 2-entry, we loose at most n such colours, preconditioned $\mathrm{rd}(K_{2^n}) = 3^n - 2^n$ being best possible. On the other hand if, we add a new vertex to an, by assumption, optimal coloured real data K_{2^n} representing genotypes of length n, then it will differ in at least one position in with every other genotype. (Assume that we extend every "old" genotype with one new position.) But then the incident edges of this new vertex are all distinct, even from the "old" edges. So we have 2^n new colours.

And the work goes on and on

In the present thesis we took a closer look at the Minimum Rainbow Subgraph problem. Starting with a short introduction to its biological roots and first approaches from the bioinformatics, we translated the problem into mathematics and developed a graph theoretical model, the Minimum Rainbow Subgraph problem. It turned out that, beside the original problem (given a set $\mathcal{G}$ of genotypes, find a set $\mathcal{H}$ of haplotypes of minimum cardinality such that $\mathcal{H}$ explains $\mathcal{G}$), the more general problem (given a p-edge-coloured graph G, find a subgraph $F \subset G$ of minimum order and size p such that every colour occurs once in F) is hard to attack but, nevertheless, very interesting to work with.

After some observations concerning the complexity and approximability, we addicted the main part of the thesis to present several algorithmic approaches tackling the problem. We discussed three different greedy algorithms (Greedy star, MRS matching and MRS_k) in terms of their running time and approximation ratio. Furthermore, the complexity consideration made for Greedy star led to results, which could be used to observe guarantees for the maximum order of solutions found by several other heuristics. We continued with the local colour density, a measure for possibly good seed of small rainbow subgraphs, and developed algorithms using this argument of choice. The final part of this chapter was dedicated to a family of algorithms, called MaxNewColour. Using the same algorithmic body, we tried several approaches in order to chose "good" new vertices. The following Table 5.2 provides an overview of the algorithms and their run-time complexity and approximation ratio.

These results show, that the Minimum Rainbow Subgraph problem is indeed difficult to handle and even hard to approximate. More efforts must be taken to find better and faster algorithms. One promising approach for future work may be taking a closer look at the so-called real data graphs, as introduced in Chapter 5. We presented some first remarks on their special

Tab. 5.2.: comparison of the discussed algorithms

algorithm	complexity	ratio
Greedy star	pn	$\frac{\Delta}{2} + \frac{\ln \Delta + 1}{2}$
MRS matching	$np^2 + p^{\frac{5}{2}}$	$\frac{3}{4}\Delta + \frac{1}{2(\Delta+1)}$
MRS_k	pk^2n^k	$\frac{1}{2} + \frac{pk}{(k-1)(\Delta+1)}$
SortLCD	p^2n^2	$\sqrt{2p}$
LCD_k	$p^3k^2n^k$	$\frac{\Delta}{2} + \frac{\ln \Delta + 1}{2}$
MaxNewColour	p^3	$\sqrt{2p}$
MaxNewColourDelta	$p^3 + np \cdot \ln np$	$\frac{\Delta}{2} + \frac{\ln \Delta + 1}{2}$
MaxNewColourFH	$np + p^3$	$\sqrt{2p}$
MaxNewColourBunt	$p^3 + np \cdot \ln np$	$\frac{\Delta}{2} + \frac{\Delta}{2(\Delta+1)}$

structure, but more investigation is needed in order to develop algorithms fitting these special colourings. Maybe a first step would be to tackle Conjecture 1. The advantage might be that small dense subgraphs guarantee a larger number of distinct colours than arbitrary subgraphs of the same order. Here "dense" is being conceived as complete. But in general even the expensive lcd value, the local colour density of Chapter 4, might be a powerful instrument worth further consideration.

On the other hand, more work has to be done in order to improve the algorithms for general graphs. Without any information on special substructures or the like it seems hard to find fast algorithms guaranteeing an approximation ratio near $1 + \varepsilon$. One of the properties, speed or accuracy, has to be relaxed in order to strengthen the other.

Another possible approach is the search for other graph classes reducing the hardness of the MRS problem. We showed that even a maximum degree of $\Delta = 2$ forced NP-hardness. So this seems a quite challenging task, too, but nevertheless worth trying.

Concluding, this thesis is only an introduction, a first step into the so far hardly explored world of minimum rainbow subgraphs. We were able to provide some basic concepts and to initially check the boundaries of the problem. Until now a lot remains unseen, a lot has to be uncovered and proved.

Bibliography

[1] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph, *Haplotyping as perfect phylogeny: a direct approach*, Journal of Computational Biology **10** (2003), no. 3-4, 323-340. ↑7, 10

[2] P. Berman and M. Karpinksi, *On some tighter inapproximability results*, Proc. of the 26th. Int. colloquium on Automata, Languages and Programming, in Lecture Notes in Comput. Sci., Vol. 1644, Springer, Berlin, 1999, pp. 100-209. ↑21

[3] P. Bonizzoni, G. Della Vedova, R. Dondi, and J. Li, *The haplotyping problem: an overview of computational models and solutions*, J. Comput. Sci. Technol. **18** (2003), no. 6, 675-688. ↑11

[4] N. Christofides, *Worst-case analysis of a new heuristioc for the travelling salesman problem*, Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976. ↑6

[5] A. Clark, *Inference of haplotypes from PCR-amplified samples of diploid populations*, Molecular Biology and Evolution **7** (1990), 111-122. ↑11

[6] J. Edmonds, *Paths, trees, and flowers*, Cand. J. Math. **17** (1965), 449-467. ↑14

[7] E. Eskin, E. Halperin, and R. Karp, *Efficient reconstruction of haplotypestructure via perfect phylogeny*, Journal of Bioinformatics and Computational Biology **1** (2003), no. 1, 1-20. ↑7, 10

[8] L. Excoffier and M. Slatkin, *Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population*, Molecular Biology and Evolution **12** (1995), no. 5, 921-927. ↑14

[9] D. Gusfield, *A practical algorithm for optimal inference of haplotypes from diploid populations*, Proceedings of the Annual International Conference on Intelligent Systems for Molecular Biology (USMB) (R. Altman, T.L. Bailey, P. Bourne, M. Gribskov, T. Lengauer, I.N. Shinyalov, L.F. Ten Eyck, and H. Weissig, eds.), AAAI Press, Menlo Park, CA, 2000, pp. 183-189. ↑11

[10] D. Gusfield, *Inference of haplotypes from samples of diploid populations: Complexity and algorithms*, Journal of Computational Biology **8** (2001), no. 3, 305-324. ↑7, 11

[11] D. Gusfield, *Haplotype inference by pure parsimony*, Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM), Lecture Notes in Computer Science, vol. 2676, 2003, pp. 144-155. ↑11

[12] F. Heinicke, *Praktikumsbericht - Heuristiken für das Regenbogenproblem*, Hochschule Mittweida, TU Bergakademie Freiberg, 2009. ↑39, 41, 42, 43, 45

[13] J.E. Hopcroft and R.M. Karp, *An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*, SIAM J. Comput. **2** (1973), 225-231. ↑14

[14] Y.-T. Huang, K.-M. Chao, and T. Chen, *An approximation algorithm for haplotype inference by maximum parsimony*, J. Comput. Biol. **12** (2005), no. 10, 1261-1274. ↑7, 14

[15] E. Hubbel, *Finding a Maximum Parsimony Solution to haplotype Phase is NP-hard*, 2001, personal communication. ↑11, 13

[16] The International HapMap Consortium, *A Haplotype Map of the Human Genome*, Nature **437** (2005), 1299-1320. ↑9

[17] J. Katrenič and I. Schiermeyer, *Improved approximation bounds for the minimum rainbow subgraph problem*, Information Processing Letters **111** (2011), 110-114. ↑21, 36, 37, 38

[18] M. Koch, *Das Population Haplotyping Problem: Graphentheoretische Ansätze*, TU Bergakademie Freiberg, 2008. ↑23, 24, 25

[19] M. Koch, S. Matos Camacho, and I. Schiermeyer, *Algorithmic approaches for the minimum rainbow subgraph problem*, Elec. Notes in Disc. Math. **38** (2011), 765-770. ↑23, 24

[20] D. König, *Graphs and matrices*, Mat. Fiz. Lapok **38** (1931), 116-119 (Hungarian). ↑14

[21] A. Kotzig, *From the theory of finite regular graphs of degree three and four*, Časopis Peětov. Mat. **82** (1957), 76-92. ↑32

[22] G. Lancia, C.M. Pinotti, and R. Rizzi, *Haplotyping populations by Pure Parsimony: Complexity of Exact and Approximation Algorithms*, INFORMS Journal on Computing **16** (2004), no. 4, 348-359. ↑6, 9, 10, 12, 13, 14

[23] G. Lancia and R. Rizzi, *A polynomial case of the parsimony haplotyping problem*, Oper. Res. Lett. **34** (2006), 289-295. ↑14

[24] S. Matos Camacho, I. Schiermeyer, and Z. Tuza, *Approximation algorithms for the minimum rainbow subgraph problem*, Disc. Math. **310** (2010), 2666-2670. ↑21, 22, 26, 31, 32

[25] S. Micali and V.V. Vazirani, *An $\mathcal{O}(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs*, Proc. Twenty-first Annual Symposium on the foundations of Computer Science (FOCS), 1980, pp. 17-21. ↑35

[26] R. Sharan, B.V. Halldorsson, and S. Istrall, *Islands of Tractability for Parsimony Haplotyping*, IEEE/ACM Transactions on Computational Biology and Bioinformatics **3** (2006), no. 3, 303-311. ↑17, 18

[27] R.S. Wang, X.S.Zhang, and L. Sheng, *Haplotype inference by pure parsimony via genetic algorithm*, Lecture notes on Operation Reasearch (ISORA2005) **5** (2005), 308-318. ↑14

[28] D.B. West, *Introduction to Graph Theory*, 2nd, 2001. ↑16