TECHNISCHE UNIVERSITÄT
CHEMNITZ

Technische Universität Chemnitz
Fakultät für Informatik
Professur Theoretische Informatik (Informationssicherheit)

Master's Thesis

Proving an Extremal Graph Coloring theorem using Linear Programming

by

Andy Oertel

Supervisors:

Prof. Dr. Hanno Lefmann
Dr. Knut Odermann

Chemnitz, May 7, 2021

Preface

Parts of this thesis were done in collaboration with Josefran de Oliveira Bastos, Carlos Hoppen, Hanno Lefmann and Dionatan Ricardo Schmidt. Together we also published our results in [Bas+21]. My contribution to [Bas+21] was primarily to obtain the Color Stability Property and especially the linear programming part by modeling and solving the linear programs. Therefore, this thesis focuses on this part in more detail and presents some more results and ideas about the linear programming. Also some additional observations and findings about the general problem are presented in this thesis.

So I would like to thank Josefran de Oliveira Bastos, Carlos Hoppen, Hanno Lefmann and Dionatan Ricardo Schmidt for working with me on this problem. Furthermore, I would like to thank Hanno Lefmann and Knut Odermann for their supervision and helpful feedback on this thesis.

Contents

1. Introduction **4**

2. Basics and Prerequisites **5**
2.1. Extremal Graph Theory 5
2.2. Edge Coloring of Graphs 7
2.3. Szemerédi Regularity Lemma 8
2.4. Graph Stability 10
2.5. Entropy Function 12
2.6. Linear Programming 14

3. The Extremal Graph Coloring Problem **14**
3.1. The Extremal Coloring Problem 15
3.2. Calculating the Upper Bound 16
3.3. Calculating the Lower Bound 21

4. The Color Stability Property **24**

5. Investigation of the Linear Program **30**
5.1. The Active Constraints of a Linear Program 31
5.2. Modifications to the Linear Program 33
5.2.1. Generalizing Proposition 4.3 33
5.2.2. Generalizing Proposition 4.4 37
5.2.3. Forbidden Complete Subgraph Constraint 41

6. Conclusion and Open Questions **47**

References **49**

Appendix A. Source Code to solve the Linear Programs **51**
A.1. Main Function Source Code 51
A.2. Linear Program (17) Source Code 52
A.3. Linear Program (29) Source Code 55
A.4. Linear Program (32) Source Code 57
A.5. Linear Program (34) Source Code 60
A.6. Linear Program (35) Source Code 62
A.7. Linear Program (37) Source Code 65
A.8. Linear Program (42) Source Code 69
A.9. Linear Program (42) Source Code with fewer constraints 71
A.10. Linear Program (45) Source Code 73

1. Introduction

The problem considered in this thesis comes from the field of extremal graph theory, which is a branch of graph theory. A well-known type of question in extremal graph theory is to ask for the maximum number of edges a graph can contain such that the graph does not contain a given subgraph. This problem was solved by Turán [Tur41] in 1941 for the complete subgraph on at least 3 vertices by showing that the balanced complete $(k-1)$-partite graph on n vertices contains the most edges without containing a complete subgraph on k vertices. This graph is also called the Turán graph $T_{k-1}(n)$.

In 1974 Erdős and Rothschild [Erd74] modified this question to ask for the maximum number of r-edge-colorings of a graph such that the colorings do not contain a monochromatic complete subgraph. For $k \geq 3$, they conjectured that the Turán graph $T_{k-1}(n)$ is the graph that admits the most 2-colorings without a monochromatic complete subgraph on k vertices. This conjecture was proven by Yuster [Yus96] for $k = 3$ and $n \geq 6$, and Alon, Balogh, Keevash and Sudakov [Alo+04] showed that the full conjecture is true for n large, but they also showed that the $T_{k-1}(n)$ cannot have the most colorings for $r \geq 4$.

This problem may be generalized to an extremal graph coloring problem where a given color pattern of a fixed graph is avoided. The extremal graph coloring problem for various fixed graphs and color patterns has already been considered by [HLO15; BHS17; HLN]. In this thesis the focus is on the problem that asks for the maximum number of r-edge-colorings without a non-rainbow colored complete subgraph on $k \geq 3$ vertices. The aim of this thesis is to prove a theorem that solves this extremal graph coloring problem and then to further improve this result. Furthermore, it would be interesting to know which graph allows that many colorings. In Section 3 this problem is studied in detail and approaches to solve parts of this problem are presented.

The following two main statements are proven. Theorem 3.7 completely solves this problem for some values of r and shows that the Turán graph $T_{k-1}(n)$ is the graph with the most colorings without containing a non-rainbow colored complete subgraph on k vertices for $k \geq 3$. However, if r is big enough, it is shown in Section 3.2 that the $T_{k-1}(n)$ cannot be the graph that admits the most r-edge-colorings and a lower bound on the number of distinct colorings is presented.

The proof of Theorem 3.7 requires the Color Stability Property provided by Lemma 4.1. Section 4 is dedicated to the proof of Lemma 4.1. The proof uses the Szemerédi Regularity Lemma [Sze78], which is a widely used tool in extremal graph theory, and a linear program solved by an algorithmic linear program solver. So the result of Lemma 4.1 depends in part on the solution of a computer program.

The value of r, for which Theorem 3.7 holds, depends on the linear program in the proof of Lemma 4.1 and it would be good if Theorem 3.7 holds for as many values of r as possible. Thus, in Section 5 the linear program used in the proof of Lemma 4.1 is analyzed. Furthermore, different approaches to improve the number of values of r for which Lemma 4.1 holds are tried. Some of these approaches even succeed or provide good insight into the problem of increasing the number of values of r. At the end of Section 5 an idea is explored that could improve the result of Theorem 3.7 even further.

Section 2 introduces basic knowledge and tools that will be helpful and necessary in the course of the thesis. Also, extremal graph theory and the basic problem of extremal graph theory, as discussed in this thesis, are also introduced there in detail.

2. Basics and Prerequisites

Before getting to the main part, the basic notation and tools are introduced, which will be used later. First, some basics about graph theory are introduced and later the tools required to solve the main problem of this thesis are presented.

Since this thesis is a part of graph theory, it is first defined what a graph is in the context of this work.

Definition 2.1 (Graph)**.** *A graph is a pair $G = (V, E)$ consisting of a vertex set V, which is finite, and an edge set E, whose elements are 2-element subsets of the vertex set.*

Sometimes the structure defined in Definition 2.1 is called an undirected finite graph, as the vertex set is finite and the edge set consist of 2-element subsets of the vertex set.

For a graph G, the vertex set of G is denoted by $V(G)$ and the edge set of G is denoted by $E(G)$. The number of edges of G is denoted by $e(G)$. Alternatively, if $V' \subseteq V(G)$, then $e(V')$ denotes the number of edges $\{u, v\} \in E(G)$ with $u, v \in V'$. For a vertex $v \in V(G)$, the set of vertices adjacent to v is the neighborhood $N(v)$. The two vertices of an edge are also referred to as the endpoints of the edge.

For $r \in \mathbb{N}$, the set $\{1, 2, \ldots, r\}$ is denoted by $[r]$.

The graph theory basics in this thesis are based on [Die05] and further information can be found there.

The Sections 2.1 and 2.2 go into more detail on the required areas of graph theory.

2.1. Extremal Graph Theory

Extremal graph theory is a branch of graph theory that encompasses the study of how global graph properties, such as the chromatic number, can influence the local substructure of a graph. In this section, the focus is on that part of extremal graph theory which asks for the maximum number of edges in a graph with a forbidden subgraph. Some results from this area, which are important for this thesis, are presented. Thus, the studied substructure in this section is a subgraph. From Section 2.2 the focus is on that part of extremal graph theory which asks for the maximum number of r-edge-colorings of a graph with a forbidden color pattern, so this section gives context and basic knowledge.

Definition 2.2 (Subgraph)**.** *A subgraph $F = (V', E')$ of a graph $G = (V, E)$ is a graph with the vertex set $V' \subseteq V$ and the edge set $E' \subseteq E$, where all edges $\{u, v\} \in E'$ satisfy $u, v \in V'$.*

In particular, complete graphs are studied as subgraphs.

Definition 2.3 (Complete graph)**.** *A complete graph $K_n = (V, E)$ is a graph on n vertices ($|V| = n$) and the edge set E consists of all possible 2-element subsets of V.*

The first problem asks for the maximum number of edges a graph G on n vertices can have where the triangle K_3 is forbidden as a subgraph of G, which means that there cannot be a subgraph of G which is a K_3. Such a graph G is called triangle-free. Basically, this means that no 3-element subset of $V(G)$ can have edges between all its vertices. This was solved by Mantel in 1907 by showing Theorem 2.4.

Theorem 2.4 (Mantel's Theorem [Man07])**.** *The maximum number of edges in a triangle-free graph on n vertices is*

$$\left\lfloor \frac{n^2}{4} \right\rfloor.$$

The only graph on n vertices that is triangle-free and has $\lfloor n^2/4 \rfloor$ edges has a vertex set V that is separated into two disjoint subsets V_1 and V_2 with $||V_1| - |V_2|| \leq 1$ and $\{u, v\} \in E$ iff $u \in V_1$ and $v \in V_2$.

Theorem 2.4 was generalized by Turán in 1941 by considering the complete graph K_k with $k \geq 2$ as the forbidden subgraph. A graph that has no K_k as a subgraph is called K_k-free. Again, the question arises about the maximum number of edges in a K_k-free graph.

Theorem 2.5 (Turán's Theorem [Tur41; Die05])**.** *For all integers n and $k \geq 2$, let G be a K_k-free graph on n vertices. Then the number of edges in G is at most*

$$\frac{k-2}{k-1} \cdot \frac{n^2}{2}.$$

Moreover, the only graph on n vertices that is K_k-free and has the most edges is the $(k-1)$-partite Turán graph $T_{k-1}(n)$.

The Turán graph is defined in Definition 2.7 and an example Turán graph is shown in Figure 1. This requires the definition of $(k-1)$-partite graphs, which are K_k-free.

Definition 2.6 (k-partite graph)**.** *For a positive integer k, a graph $G = (V, E)$ is called k-partite if V can be partitioned into k disjoint subsets $V_1, \ldots, V_k$, where every edge $\{u, v\} \in E$ satisfies $u \in V_i, v \in V_j$ for $i \neq j$. The subsets $V_1, \ldots, V_k$ are called classes.*

The $(k-1)$-partite graph on n vertices with the most edges is the Turán graph $T_{k-1}(n)$.

Definition 2.7 (Turán graph)**.** *For $k \geq 2$, the Turán graph $T_{k-1}(n) = (V, E)$ is a graph on n vertices. The vertex set is partitioned into $(k-1)$ disjoint subsets $V_1, \ldots, V_{k-1}$, where $||V_i| - |V_j|| \leq 1$ for $1 \leq i < j \leq (k-1)$. An edge u, v is in E iff $u \in V_i$ and $v \in V_j$ for all $i, j \in [k-1], i \neq j$.*

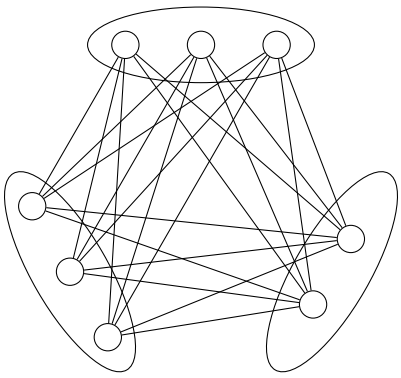


Figure 1: The 3-partite Turán graph $T_3(8)$ on 8 vertices as an example of a Turán graph.

The number of edges of the edge maximal K_k-free graph on n vertices is called the extremal number and is denoted by $\mathrm{ex}(n, K_k)$. Therefore, $\mathrm{ex}(n, K_k)$ is the tight upper bound on the number of edges a K_k-free graph on n vertices can have, whereas the upper bound of Theorem 2.5 is not necessarily tight.

By Theorem 2.5, the Turán graph is K_k-extremal, thus the definition of the $T_{k-1}(n)$ is used to calculate the exact value of $\mathrm{ex}(n, K_k)$. The exact value of $\mathrm{ex}(n, K_k)$ is

$$\begin{aligned} &\binom{n}{2} - \binom{\lfloor n/(k-1) \rfloor}{2} \left(\left\lfloor \frac{n}{k-1} \right\rfloor (k-1) - n + (k-1) \right) \\ &\quad - \binom{\lceil n/(k-1) \rceil}{2} \left(n - \left\lfloor \frac{n}{k-1} \right\rfloor (k-1) \right). \end{aligned} \tag{1}$$

This statement follows from subtracting the number of missing edges within the classes from the number of edges of the K_n. The number of edges with both endpoints in the same class of size j is $\binom{j}{2}$. When dividing the vertex set into $(k-1)$ classes of about the same size, there are

$$\left(\left\lfloor \frac{n}{k-1} \right\rfloor (k-1) - n + (k-1)\right)$$

smaller classes of size $\lfloor n/(k-1) \rfloor$ and

$$\left(n - \left\lfloor \frac{n}{k-1} \right\rfloor (k-1)\right)$$

larger classes of size $\lceil n/(k-1) \rceil$.

Known bounds on the extremal number $\mathrm{ex}(n, K_k)$ are

$$\frac{(k-2)n^2}{(k-1)2} - (k-1) < \mathrm{ex}(n, K_k) \leq \frac{(k-2)n^2}{(k-1)2}. \tag{2}$$

This problem can be further generalized for a forbidden arbitrary fixed graph F, but this is not considered further in this thesis. This generalization was solved by Erdős and Stone [ES46] for non-bipartite F and more details about this problem can also be found in [Die05].

2.2. Edge Coloring of Graphs

At the beginning of this section the concept of edge coloring is introduced and then used as the forbidden substructure for an extremal graph theory problem. First, the r-edge-coloring of a graph is defined. In this thesis, only the coloring of the edges is considered, and therefore r-coloring is used as a short form for r-edge-coloring.

Definition 2.8 (r-edge-coloring (r-coloring))**.** *An r-edge-coloring of a graph $G = (V, E)$ is a map $C_{r,G} : E \to [r]$, where the elements of $[r]$ could be viewed as colors or classes.*

The r-edge-coloring of a graph could also be viewed as a partitioning of the edge set E into r disjoint subsets $E_1 \dot\cup \ldots \dot\cup E_r = E$.

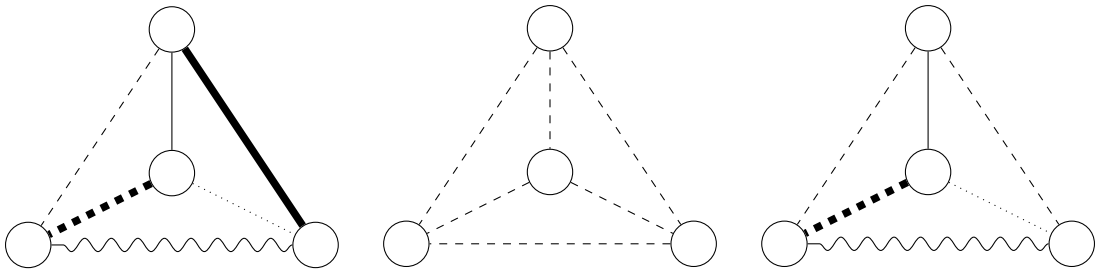
The r-colorings of a fixed graph F can be grouped by so-called patterns P into sets of r-colorings of F. These sets are called color patterns (F, P). A well-known pattern is the rainbow pattern R, where all edges of F are colored with a different color. Another pattern is the non-rainbow pattern $\overline{R}$, where at least two edges of F are colored with the same color. Figure 2 illustrates some example colorings of the K_4 colored according to these two patterns.

Similar to Section 2.1, the color pattern (F, P) is now studied as the forbidden substructure. An r-coloring of a graph G is (F, P)-free if G does not contain a copy of F that is colored with pattern P. This means, G may still contain F as a subgraph, but this subgraph cannot be colored in accordance with pattern P. Therefore, the property that a r-coloring of G is (F, P)-free is a property of the coloring.

The set of all (F, P)-free r-colorings of a graph G is denoted by $\mathcal{C}_{r,(F,P)}(G)$. Furthermore, the maximum number of (F, P)-free r-colorings a graph on n vertices can have is

$$c_{r,(F,P)}(n) = \max\{|\mathcal{C}_{r,(F,P)}(G)| : |V(G)| = n\}.$$

The graph G on n vertices is called $\mathcal{C}_{r,(F,P)}$-extremal iff $|\mathcal{C}_{r,(F,P)}(G)| = c_{r,(F,P)}(n)$. Finding $c_{r,(F,P)}(n)$ and the corresponding $\mathcal{C}_{r,(F,P)}$-extremal graph is the extremal graph coloring



(a) Color pattern (K_4, R). (b) Color pattern $(K_4, \overline{R})$. (c) Color pattern $(K_4, \overline{R})$.

Figure 2: Example colorings of the K_4, which is colored according to the rainbow and non-rainbow pattern. The figures (b) and (c) are two examples of the same color pattern $(K_4, \overline{R})$.

problem that will be studied in this thesis. Especially, the non-rainbow pattern of the complete graph $(K_k, \overline{R})$ for $k \geq 3$ is studied.

In Section 3, the extremal graph coloring problem will be introduced in more detail and an approach to solve this problem for some values of r is presented. Before it is possible to investigate the extremal coloring problem further, necessary tools are introduced.

2.3. Szemerédi Regularity Lemma

The Szemerédi regularity lemma is a widely used tool in extremal graph theory and it is also used in the proof to solve the extremal graph coloring problem considered in this thesis. The original lemma was proven by Endre Szemerédi [Sze78] in 1978.

To state the regularity lemma, some definitions are needed. For a graph $G = (V, E)$ and disjoint subsets $A, B \subseteq V$, the number of edges with one endpoint in A and the other endpoint in B is denoted by $e(A, B)$. The edge-density between two subset of the vertex set is defined as follows.

Definition 2.9 (Edge-density)**.** *Let $G = (V, E)$ be a graph and let $A, B \subseteq V$ be disjoint and non-empty, the edge-density $d(A, B)$ between A and B is defined as*

$$d(A, B) \coloneqq \frac{e(A, B)}{|A| \cdot |B|}.$$

This makes the definition of ε-regularity possible.

Definition 2.10 (ε-regular pair)**.** *Let $G = (V, E)$ be a graph and $A, B \subseteq V$ be disjoint and non-empty. For $\varepsilon > 0$, the pair (A, B) is called ε-regular if all $X \subseteq A$ and $Y \subseteq B$ with $|X| > \varepsilon|A|$ and $|Y| > \varepsilon|B|$ satisfy*

$$|d(X, Y) - d(A, B)| < \varepsilon.$$

To extend the concept of ε-regularity, the balanced partition of a graph is defined.

Definition 2.11 (Balanced partition)**.** *The balanced partition of the set V partitions V into k disjoint subsets $V_1 \dot\cup \ldots \dot\cup V_k = V$, where $||V_i| - |V_j|| \leq 1$ for all $i, j \in [k], i \neq j$.*

The ε-regularity of a balanced partition of a graph is defined as follows.

Definition 2.12 (ε-regular partition)**.** *Let $G = (V, E)$ be a graph and $\varepsilon > 0$. A balanced partition of the set V into $V_1, \ldots, V_m$ is called ε-regular if at most $\varepsilon\binom{m}{2}$ of the pairs (V_i, V_j) are not ε-regular for all $i, j \in [m], i \neq j$.*

This enables to state the regularity lemma by Szemerédi [Sze78].

Lemma 2.13 (Szemerédi Regularity Lemma [Die05; Alo+04])**.** *For each $\varepsilon > 0$, there exists an integer $M = M(\varepsilon) > 0$ such that every graph $G = (V, E)$ with at least $n > M$ vertices admits an ε-regular partition of V into $V_1, \dots, V_m$, for some $1/\varepsilon \leq m \leq M$.*

Lemma 2.13 can also be adopted to the r-edge-coloring of a graph, which can also be found in [KS96] and this requires the definition of multicolored ε-regularity.

Definition 2.14 (Multicolored ε-regular partition)**.** *Let $G = (V, E)$ be a graph and $\varepsilon > 0$. The edge set is r-colored and according to the colors partitioned into $E = E_1 \dot\cup \dots \dot\cup E_r$. Then the balanced partition of the set V into $V_1, \dots, V_m$ is called multicolored ε-regular if the partition is ε-regular simultaneously with respect to the graphs $G_i = (V, E_i)$ for all $i \in [r]$.*

The multicolored regularity lemma can now be stated.

Lemma 2.15 (Multicolored Regularity Lemma [KS96])**.** *For each $\varepsilon > 0$ and integer r, there exists an integer $M = M(\varepsilon, r) > 0$ such that if the edges of a graph $G = (V, E)$ on $n > M$ vertices are r-colored then there exists a multicolored ε-regular partition of V into $V_1, \dots, V_m$, for some $1/\varepsilon \leq m \leq M$.*

Associated with the multicolored ε-regular partition, the multicolored cluster graph can be defined.

Definition 2.16 (Multicolored cluster graph)**.** *Let G be a graph, $\varepsilon > 0$ and $\eta > 0$. For a multicolored ε-regular partition of $V(G)$ into $V_1, \dots, V_m$, the multicolored cluster graph $\mathcal{H}(\eta)$ is a graph with the vertex set $[m]$, where the vertex i represents the class V_i, and $e = \{i, j\}$ is an edge of $\mathcal{H}(\eta)$ if the pair (V_i, V_j) is ε-regular in G for every color and the edge-density $d(V_i, V_j)$ is at least η for some color $c \in [r]$. Each edge $e \in E(\mathcal{H})$ is assigned a list L_e containing all colors for which the edge-density is at least η.*

With the definition of the multicolored cluster graph, the following embedding lemma can be stated. The proof of this result follows from arguments in the proof of Theorem 2.1 by Komolós and Simonovits [KS96].

Lemma 2.17 (Embedding Lemma by Theorem 2.1 of [KS96])**.** *For $\eta > 0$ and positive integers k and r, there exists $\varepsilon = \varepsilon(r, \eta, k) > 0$ and a positive integer $n_0 = n_0(r, \eta, k)$ with the following property. Suppose that $C_{r,G}$ is an r-coloring of the graph G on $n > n_0$ vertices with a multicolored ε-regular partition $V_1, \dots, V_m$ that defines the multicolored cluster graph $\mathcal{H} = \mathcal{H}(\eta)$. Let F be a fixed graph on k vertices colored with pattern P on $t \leq r$ colors. If $\mathcal{H}$ contains (F, P), then the coloring $C_{r,G}$ also contains (F, P).*

To prove the theorem that solves the extremal graph coloring problem, the following auxiliary lemma is used.

Lemma 2.18. *Let $r \geq 1$ and $k \geq 2$ be positive integers and let C_{r,K_k} be an r-coloring of the K_k with vertex set $\{v_1, \dots, v_k\}$ such that each edge $\{v_i, v_j\}$ has color $\alpha_{i,j} \in [r]$. Let $\omega : [k] \to (0, 1]$, with $\omega(i) \leq 1/(i-1)$, be a non-increasing function and fix*

$$\beta \geq \max_{1<i\leq k} \omega(i)/\omega(i-1).$$

Let $C_{r,G}$ be an r-coloring of a graph G whose vertex set contains mutually disjoint sets $W_1, \dots, W_k$ with the following property. For every pair $\{i, j\} \subseteq [k]$ and all subsets $X_i \subseteq W_i$, where $|X_i| \geq \omega(k)|W_i|$, and $X_j \subseteq W_j$, where $|X_j| \geq \omega(k)|W_j|$, there are at least $\beta|X_i||X_j|$ edges of color $\alpha_{i,j}$ between X_i and X_j in $C_{r,G}$. Then $C_{r,G}$ contains a copy of C_{r,K_k} with one vertex in each set W_i.

Proof. Fix an integer $r > 0$. The proof is by induction over k. For $k = 2$ the result is trivial, as $\beta|X_1||X_2| > 0$, thus there is at least one edge of color $\alpha_{1,2}$ with endpoints in X_1 and X_2. For the induction step, fix $k \geq 3$ and let C_{r,K_k}, $C_{r,G}$, ω and β be as in the statement of the lemma. By induction, suppose that the statement holds.

For each $i \in [k-1]$, let $W_k^{\alpha_{i,k}} \subseteq W_k$ be the maximal subset of W_k such that each vertex of $W_k^{\alpha_{i,k}}$ has fewer than $\beta|W_i|$ neighbors in $|W_i|$ whose edges have color $\alpha_{i,k}$. So there are fewer than $\beta|W_i||W_k^{\alpha_{i,k}}|$ edges of color $\alpha_{i,k}$ between W_i and $W_k^{\alpha_{i,k}}$. By the definition of $C_{r,G}$, it holds that $|W_k^{\alpha_{i,k}}| < \omega(k)|W_k|$, as otherwise at least $\beta|W_i||W_k^{\alpha_{i,k}}|$ edges of color $\alpha_{i,k}$ would be between W_i and $W_k^{\alpha_{i,k}}$. Since $\omega(k) \leq 1/(k-1)$, it holds that

$$\left|\bigcup_{i=1}^{k-1} W_k^{\alpha_{i,k}}\right| \leq \sum_{i=1}^{k-1} |W_k^{\alpha_{i,k}}| < |W_k|.$$

Therefore, there exists a vertex $x_k \in W_k$ that has at least $\beta|W_i|$ neighbors in W_i for every $i \in [k-1]$ whose edges have color $\alpha_{i,k}$, as $x_k \notin \bigcup_{i=1}^{k-1} W_k^{\alpha_{i,k}}$.

Set $W_i' = N(x_k) \cap W_i$. Let $C_{r,G'}$ be an r-coloring of the subgraph of G with vertex set $V(G') = \bigcup_{i=1}^{k-1} W_i'$ and the same r-coloring as $C_{r,G}$ and let $C_{r,K_{k-1}}$ be an r-coloring of the subgraph of the K_k with vertex set $V(K_{k-1}) = \{v_1, \ldots, v_{k-1}\}$ and the same r-coloring as C_{r,K_k}. It will be shown that induction can be applied on $C_{r,G'}$ and $C_{r,K_{k-1}}$ to obtain vertices $x_1, \ldots, x_{k-1}$ with $x_i \in W_i$. This generates $C_{r,G''}$ of the subgraph with vertex set $V(G'') = \{x_1, \ldots, x_k\}$ and the same coloring as $C_{r,G}$ that is a copy of the coloring C_{r,K_k}. By definition, if ω and β satisfy the conditions of the lemma for k, then β and ω also satisfy the conditions for $k-1$. Furthermore, for any fixed $X_i \subseteq W_i'$, with $|X_i| \geq \omega(k-1)|W_i'|$, it holds that

$$|X_i| \geq \omega(k-1)|W_i'| \geq \omega(k-1)\beta|W_i| \geq \omega(k)|W_i|.$$

Thus, by the choice of $C_{r,G}$, all pairs of sets $X_i \subseteq W_i'$ with $|X_i| \geq \omega(k-1)|W_i'|$ and $X_j \subseteq W_j'$ with $|X_j| \geq \omega(k-1)|W_j'|$ have at least $\beta|X_i||X_j|$ edges of color $\alpha_{i,j}$ between X_i and X_j. This concludes that induction can be applied and the lemma follows directly. □

2.4. Graph Stability

Stability results for graphs are also a widely used tools to solve problems in the area of extermal graph theory. The following theorem by Füredi is used. The proof of Theorem 2.19 can be found in [Für15].

Theorem 2.19 (Theorem 2 of [Für15])**.** *Let t be a positive integer and $G = (V, E)$ be a K_k-free graph on n vertices. If $|E| = \mathrm{ex}(n, K_k) - t$, then there exists a partition $V = V_1 \dot\cup \ldots \dot\cup V_{k-1}$ with $\sum_{i=1}^{k-1} e(V_i) \leq t$.*

Furthermore, the following lemma by Alon and Yuster [AY06] is used.

Lemma 2.20 (Lemma 2.3 of [AY06])**.** *Let $0 < t < \frac{n^2}{4(k-1)^2}$ and let $G = (V, E)$ be a $(k-1)$-partite graph on n vertices with partition $V = V_1 \cup \cdots \cup V_{k-1}$ and with at least $\mathrm{ex}(n, K_k) - t$ edges. If at least $(2k-1)t$ new edges are added to G, then in the resulting graph there is a K_k subgraph with exactly one new edge connecting two vertices in the same class V_i of G.*

Proof. Let $V = V_1 \cup \cdots \cup V_{k-1}$ be the partition of G. As $e(G) \geq \mathrm{ex}(n, K_k) - t > \mathrm{ex}(n, K_{k-1})$ and by Turán's Theorem [Tur41], G contains K_{k-1} subgraphs. If at least $(2k-1)t$ new edges are added to G, one class, say V_1, contains at least $2t$ of the new edges. Since every graph contains a bipartite subgraph with the same vertex set and more than half the

number of edges, a bipartite subgraph G' induced by the new edges within V_1 with more than t edges can be obtained. The sum of the number of edges in G' and the number of edges in G is larger than $\mathrm{ex}(n, K_k)$, hence there exists a K_k subgraph containing exactly one new edge with both endpoints in V_1, as G' is bipartite. □

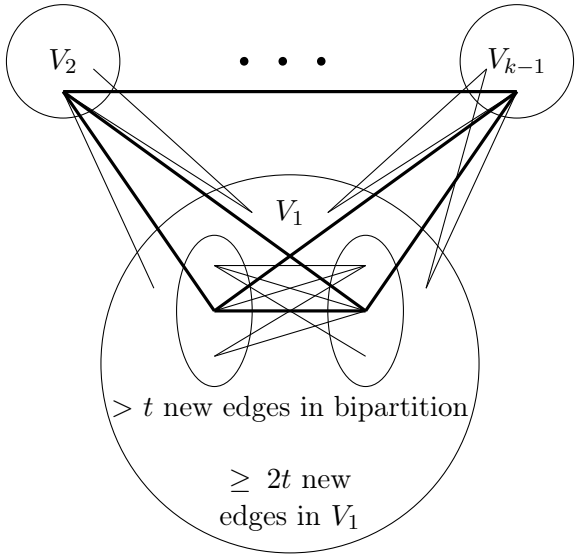


Figure 3: Sketch of the idea behind the proof of Lemma 2.20. The circles represent the classes and the ellipses the two classes of the bipartition. The bold edges show how the K_k generated by Lemma 2.20 looks like.

A sketch to illustrate the arguments in the proof of Lemma 2.20 is shown in Figure 3. The following bound on the size of the classes in a $(k-1)$-partite graph is stated for later use.

Lemma 2.21. *For integer $k \geq 3$, let $G = (V, E)$ be a $(k-1)$-partite graph on n vertices with partition $V = V_1 \cup \cdots \cup V_{k-1}$. For some $t \geq (k-1)^2$, if the graph G contains at least $\mathrm{ex}(n, K_k) - t$ edges, then for $i \in \{1, \ldots, k-1\}$ it holds that*

$$\left| |V_i| - \frac{n}{k-1} \right| < \sqrt{2t}.$$

Proof. If $|V_{k-1}| = x$, then G contains at most

$$x(n-x) + \binom{k-2}{2} \cdot \left(\frac{n-x}{k-2}\right)^2$$

edges. For the second summand, it is used that for non-negative integers $a_1, \ldots, a_{k-2}$ summing to $a_1 + \ldots + a_{k-2} = M$, the value of $\sum_{1 \leq i < j \leq k-2} a_i a_j$ is maximal for $a_1 = \cdots = a_{k-2} = M/(k-2)$.

Using (2), it holds that

$$x(n-x) + \binom{k-2}{2} \cdot \left(\frac{n-x}{k-2}\right)^2 \geq \mathrm{ex}(n, K_k) - t \geq \frac{(k-2)n^2}{2(k-1)} - (k-1) - t,$$

concluding that

$$x^2 - \frac{2}{k-1}nx + \frac{1}{(k-1)^2}n^2 \leq \frac{2(k-2)}{k-1} \cdot t + 2(k-2).$$

Thus,

$$\left| x - \frac{n}{k-1} \right| \leq \sqrt{\frac{2(k-2)}{k-1} t + 2(k-2)} < \sqrt{2t}$$

for $t \geq (k-1)^2$, as required. □

2.5. Entropy Function

For some proofs in this thesis, the binary entropy function is used in estimations. The function is defined as follows.

Definition 2.22 (Binary entropy function)**.** *The binary entropy function* $H : [0,1] \to [0,1]$ *is defined as*

$$H(x) \coloneqq \begin{cases} -x \log_2 x - (1-x) \log_2 (1-x) & , x \notin \{0,1\} \\ 0 & , x \in \{0,1\}. \end{cases}$$

The binary entropy function is illustrated in Figure 4. It is easy to see that $H(x)$ has its maximum at $x = 1/2$ and the minima are at $x = 0$ and $x = 1$.

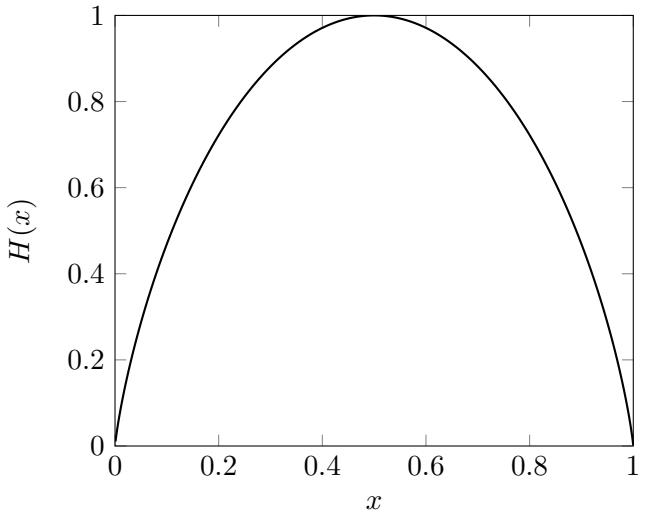


Figure 4: Binary entropy function $H(x)$.

The Lemma 2.23 is a well-known inequality to estimate the binomial coefficient.

Lemma 2.23. *Let* $H(x)$ *be the binary entropy function and* $0 \leq \alpha \leq 1$*, then for large* n

$$\binom{n}{\alpha n} \leq 2^{H(\alpha) n}. \tag{3}$$

Proof. The binomial coefficient can be written as

$$\binom{n}{\alpha n} = \frac{n!}{(\alpha n)! \cdot ((1-\alpha) n)!}. \tag{4}$$

Stirling's approximation [Rom00] is used to approximate the right side of (4). Stirling's approximation states the asymptotic equality for $n \to \infty$ is

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e} \right)^n,$$

where e is Euler's number. The calculation continues with

$$\begin{aligned}\binom{n}{\alpha n} &\sim \frac{\left(\frac{n}{e}\right)^n}{\left(\frac{\alpha n}{e}\right)^{\alpha n}\cdot\left(\frac{(1-\alpha)n}{e}\right)^{(1-\alpha)n}}\frac{\sqrt{2\pi n}}{\sqrt{2\pi\alpha n}\cdot\sqrt{2\pi(1-\alpha)n}}\\ &= \frac{\left(\frac{n}{e}\right)^n}{\left(\frac{n}{e}\right)^{(\alpha+(1-\alpha))n}\cdot\alpha^{\alpha n}\cdot(1-\alpha)^{(1-\alpha)n}}\frac{\sqrt{2\pi n}}{\sqrt{2\pi n}\cdot\sqrt{2\pi\alpha(1-\alpha)n}}\\ &= \frac{1}{\alpha^{\alpha n}\cdot(1-\alpha)^{(1-\alpha)n}}\frac{1}{\sqrt{2\pi\alpha(1-\alpha)n}}.\end{aligned}$$

Taking the binary logarithm and dividing by n results in

$$\begin{aligned}\frac{1}{n}\log_2\binom{n}{\alpha n} &\sim -(\alpha\log_2\alpha+(1-\alpha)\log_2(1-\alpha))-\frac{1}{2n}\log_2(2\pi\alpha(1-\alpha)n)\\ &= H(\alpha)-\frac{1}{2n}\log_2(2\pi\alpha(1-\alpha)n)\\ &\leq H(\alpha).\end{aligned}\tag{5}$$

Therefore, if n is large enough, (5) is

$$\frac{1}{n}\log_2\binom{n}{\alpha n}\leq H(\alpha).\tag{6}$$

The desired result is obtained by multiplying (6) by n and then exponentiating by base 2

$$\binom{n}{\alpha n}\leq 2^{H(\alpha)n}.$$

□

An estimate on the binary entropy function is stated in Lemma 2.24.

Lemma 2.24. *Let $H(x)$ be the binary entropy function, then for $x\leq 1/8$*

$$H(x)\leq -2x\log_2 x.\tag{7}$$

Proof. By using Definition 2.22 for the binary entropy function, it holds that

$$\begin{aligned}H(x)&\leq -2x\log_2 x\\ -x\log_2 x-(1-x)\log_2(1-x)&\leq -2x\log_2 x\\ f(x)\coloneqq x\log_2 x-(1-x)\log_2(1-x)&\leq 0.\end{aligned}$$

Taking the derivative of $f(x)$ gives

$$f'(x)=\frac{\ln x+\ln(1-x)+2}{\ln 2}.$$

As $f'(x)\leq 0$ for $x\leq 1/8$ and $f(1/8)<0$, it follows that $f(x)\leq 0$ for $x\leq 1/8$ and the statement (7) holds. □

2.6. Linear Programming

Linear programming is a method from the area of mathematical optimization. Since linear programming is only used as a tool in some proofs of this thesis, the linear programming problem and methods to solve such problems are only mentioned and not introduced in detail. A detailed introduction to linear programming can be found in [Van20].

A linear program, or short LP, is an optimization problem that asks for the minimum or maximum value of an objective function under given constraints. In this thesis the following form of a linear program is used.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0, \end{aligned}$$

where $x \in \mathbb{R}^n$ are the variables of the linear program, and $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. The term $c^T x$ is called the objective function and the term $Ax \leq b$ are the constraints. Generally, a possible solution $x' \in \mathbb{R}^n$ of the linear program satisfies $Ax' \leq b$. The optimal solution $x^* \in \mathbb{R}^n$ is a possible solution where for all possible solutions x' it holds that

$$c^T x^* \geq c^T x'.$$

The optimal value of a linear program is $c^T x^*$.

Alternatively a linear program can be written by explicitly writing out the matrix-vector operations, which results in the linear program

$$\begin{aligned} \max \quad & c_1 x_1 + \ldots + c_n x_n \\ \text{s.t.} \quad & \sum_{j=1}^{n} A_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \\ & x_1, \ldots, x_n \geq 0. \end{aligned}$$

The field of linear programming offers many methods to solve a linear program, which is finding the optimal value of this linear program. The two most commonly used methods are the simplex method and the interior-point method. Both methods will not be explained further, as already implemented software solutions, so called solvers, are used to solve the linear programs in this thesis.

The solutions of linear programs in this thesis are obtained by modeling the linear program in *Rust* [Dev20; 17] and solving it using the *CPLEX* [IBM19] solver. The *CPLEX* solver is configured to use the dual simplex method to obtain the optimal solution and value. Every time a solution of a linear program is obtained, the corresponding source code is referenced in Appendix A. The source code to setup the environment and solve the linear program is given in Appendix A.1 and this source code is the same for all the linear programs in this thesis.

The limiting factor for solving the linear programs in this thesis is the size of the computer memory. In this thesis, 48 GB of RAM were used to obtain the optimal solutions and it would be possible to obtain more solutions of the linear programs if more memory is used.

3. The Extremal Graph Coloring Problem

In this section, the extremal graph coloring problem, which was mentioned at the end of Section 2.2, is presented in more detail. Then the special extremal graph coloring problem

that considers the non-rainbow pattern of the complete graph is studied. Solving this special extremal graph coloring problem is the main goal of this thesis and an approach to solve this problem for some r is presented.

3.1. The Extremal Coloring Problem

As mentioned in Section 2.2, the extremal graph coloring problem asks for the value of $c_{r,(F,P)}(n)$ for positive integers n, r and a fixed color pattern (F, P). Furthermore, it would be interesting to know the graph on n vertices that has $c_{r,(F,P)}(n)$ many distinct (F, P)-free r-colorings.

A simple lower bound on $c_{r,(F,P)}(n)$ is

$$c_{r,(F,P)}(n) \geq r^{\mathrm{ex}(n,F)},$$

as an F-free graph can only be colored (F, P)-free, thus the $\mathrm{ex}(n, F)$ edges of the F-extremal graph on n vertices can be colored arbitrarily with r colors.

A trivial upper bound on $c_{r,(F,P)}(n)$ is

$$c_{r,(F,P)}(n) \leq r^{\binom{n}{2}},$$

as this is the maximum number of possible r-colorings a graph on n vertices can have without any restrictions. Hence, this upper bound is not very helpful.

Giving a general solution to the extremal graph coloring problem is difficult, so this thesis focuses only on a single type of color pattern. This special extremal graph coloring problem asks for the maximum number of $(K_k, \overline{R})$-free r-colorings a graph on n vertices can have, which is denoted by $c_{r,(K_k,\overline{R})}(n)$. In this thesis, this problem is studied for $k \geq 3$, as the problem for $k \leq 2$ is trivial. From the aforementioned bounds, it is known that the lower bound on $c_{r,(K_k,\overline{R})}(n)$ is

$$c_{r,(K_k,\overline{R})}(n) \geq r^{\mathrm{ex}(n,K_k)} \approx r^{\frac{(k-2)n^2}{2(k-1)}}.$$

The graph on n vertices that has this many $(K_k, \overline{R})$-free r-colorings is the Turán graph $T_{k-1}(n)$. Therefore, the $T_{k-1}(n)$ is a good candidate as the $(K_k, \overline{R})$-extremal graph. In Theorem 3.7 it is later proven that this graph is indeed $(K_k, \overline{R})$-extremal for relatively small r.

Other good candidates for $(K_k, \overline{R})$-extremal graphs are the Turán graphs $T_f(n)$ for $f > (k-1)$. This can be illustrated by a simple example and comparing the number of possible r-colorings.

Example 3.1. Let $r = 27$, $(K_3, \overline{R})$ be the forbidden color pattern and n be divisible by 4. The $T_2(n)$ can be colored arbitrarily with the 27 available colors. This results in

$$|\mathcal{C}_{27,(K_3,\overline{R})}(T_2(n))| = 27^{\frac{n^2}{4}}$$

possible colorings. To color the $T_4(n)$, the color set is separated into three pairwise disjoint subsets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$, where $|\mathcal{C}_1| = |\mathcal{C}_2| = |\mathcal{C}_3| = 9$. The four classes of the $T_4(n)$ are denoted by V_1, V_2, V_3, V_4. The edges of this graph are colored as follows. The edges between V_1 and V_2 and between V_3 and V_4 are colored arbitrarily with the colors in $\mathcal{C}_1$. The edges between V_1 and V_3 and between V_2 and V_4 are colored arbitrarily with the colors in $\mathcal{C}_2$. The edges between V_1 and V_4 and between V_2 and V_3 are colored arbitrarily with the colors in $\mathcal{C}_3$.

This coloring is illustrated in Figure 5. Thus, the number of possible colorings of the $T_4(n)$ is

$$|\mathcal{C}_{27,(K_3,\overline{R})}(T_4(n))| \geq 9^{\frac{3n^2}{8}}.$$

In fact, the number of colorings $|\mathcal{C}_{27,(K_3,\overline{R})}(T_4(n))|$ is larger than $9^{3n^2/8}$ because many colorings are missing. For instance, $\mathcal{C}_1$ and $\mathcal{C}_2$ could be swapped to generate more possible colorings.

This results in the $T_4(n)$ having more colorings than the $T_2(n)$, which can be seen by

$$|\mathcal{C}_{27,(K_3,\overline{R})}(T_2(n))| = 27^{\frac{n^2}{4}} = 9^{\frac{3}{2}\cdot\frac{n^2}{4}} = 9^{\frac{3n^2}{8}} < |\mathcal{C}_{27,(K_3,\overline{R})}(T_4(n))|.$$

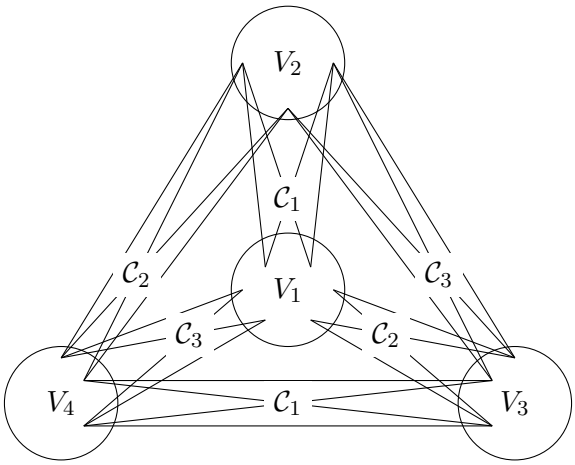


Figure 5: Illustration of the assignment of color subsets to the $T_4(n)$ as in Example 3.1. The circles represent the classes of the $T_4(n)$. All edges connecting two classes can be colored with all the colors of the color subset written over the edges.

3.2. Calculating the Upper Bound

An equivalent argument as in Example 3.1 could be applied to every K_k for $k \geq 3$. So there exists an $r_1(k)$ such that, for every $r \geq r_1(k)$ and some $f > (k-1)$, $|\mathcal{C}_{r,(K_k,\overline{R})}(T_f(n))|$ is bigger than $|\mathcal{C}_{r,(K_k,\overline{R})}(T_{k-1}(n))|$. The number $r_1(k)$ is the smallest r for which it is known that the $T_{k-1}(n)$ is not a $(K_k,\overline{R})$-extremal graph and this can be calculated with the following procedure.

For every $k \geq 4$ and $f > (k-1)$, the smallest r for which it is known that the $T_f(n)$ has more $(K_k,\overline{R})$-free r-colorings than the $T_{k-1}(n)$ is denoted as $r_1(k,f)$. To calculate $r_1(k,f)$, a general estimate of $|\mathcal{C}_{r,(K_k,\overline{R})}(T_f(n))|$ is needed. This time the color set needs to be partitioned into $\binom{f}{2}$ subsets. For each pair of classes, all the edges with one endpoint in one class and the other endpoint in the other class are assigned one of these color subsets, which is different for each pair of classes. Then the edges are colored with a color from the assigned color subset. Such a coloring is trivially $(K_k,\overline{R})$-free, as every K_k subgraph is colored rainbow. At the moment, no better method is known to estimate the number of

$(K_k, \overline{R})$-free r-colorings of the $T_f(n)$.

$$|\mathcal{C}_{r,(K_k,\overline{R})}(T_{k-1}(n))| = r^{\frac{(k-2)n^2}{2(k-1)}} \leq \left(\frac{r}{\binom{f}{2}}\right)^{\frac{(f-1)n^2}{2f}} < |\mathcal{C}_{r,(K_k,\overline{R})}(T_f(n))|$$

$$r^{\frac{k-2}{k-1}} \leq \left(\frac{r}{\binom{f}{2}}\right)^{\frac{f-1}{f}}$$

$$r^{\frac{k-f-1}{f(k-1)}} \leq \left(\frac{1}{\binom{f}{2}}\right)^{\frac{f-1}{f}}$$

$$r \geq \binom{f}{2}^{\frac{(f-1)(k-1)}{f-k+1}} =: r_1(k,f).$$

Then $r_1(k)$ can be calculated by

$$r_1(k) = \min_{f>(k-1)} r_1(k,f).$$

For $k = 3$, the term to estimate $|\mathcal{C}_{r,(K_k,\overline{R})}(T_f(n))|$ could be larger than the estimate used for $k \geq 4$, as one color subsets can be used multiple times. Proposition 3.2 shows this statement and gives the number of required color subsets.

Proposition 3.2. *For $k = 3$, $f > 2$ and f even, the $T_f(n)$ can be colored $(K_3, \overline{R})$-free using $(f-1)$ disjoint subsets of the color set.*

For $k = 3$, $f > 2$ and f odd, the $T_f(n)$ can be colored $(K_3, \overline{R})$-free using f disjoint subsets of the color set.

To prove Proposition 3.2 the maximum matching of a graph is used.

Definition 3.3 (Maximum matching)**.** *A matching M of a graph $G = (V, E)$ is a subset of the edge set E, where no two edges in M have a common vertex. A maximum matching of a graph $G = (V, E)$ is a matching of G that has the maximum number of edges. The size of the maximum matching is called matching number.*

Proof of Proposition 3.2. Let $V_1, \ldots, V_f$ be the classes of the $T_f(n)$ and let E_{ij} denote the edge subset between V_i and V_j. To avoid a non-rainbow coloring of the K_3, every K_3 must always be colored rainbow. In the $T_f(n)$ a K_3 subgraph cannot contain two edges that are from the same edge subset, as this would require edges inside the classes. Thus, the edges in the same E_{ij} could be colored with the same color and every E_{ij} could be viewed as a single edge. So the K_f is now considered as the graph to color $(K_3, \overline{R})$-free and the observations from the K_f are then adopted to the $T_f(n)$.

To color the K_f such that every K_3 subgraph is colored rainbow, two edges that have a vertex edge cannot be colored with the same color. For a single color, this condition is equal to the definition of a matching. If there exists a set of disjoint matchings that covers all edges, every matching can be assigned a different color and this would produce a $(K_3, \overline{R})$-free r-coloring of the K_f. Furthermore, every matching could be assigned a pairwise disjoint subset of colors, where the color of every edge could be chosen arbitrarily from this color subset.

To get the most colorings, the number of color subsets must be as small as possible, which means that the matching must be as big as possible. This is the case for the maximum

matching. The matching number of the K_f is $\lfloor f/2 \rfloor$. Dividing the number of edges in the K_f by the matching number results in at least

$$\frac{\frac{f(f-1)}{2}}{\left\lfloor \frac{f}{2} \right\rfloor} = \begin{cases} \frac{f(f-1)}{2} \cdot \frac{2}{f} = f-1 & \text{if } f \text{ is even} \\ \frac{f(f-1)}{2} \cdot \frac{2}{f-1} = f & \text{if } f \text{ is odd} \end{cases}$$

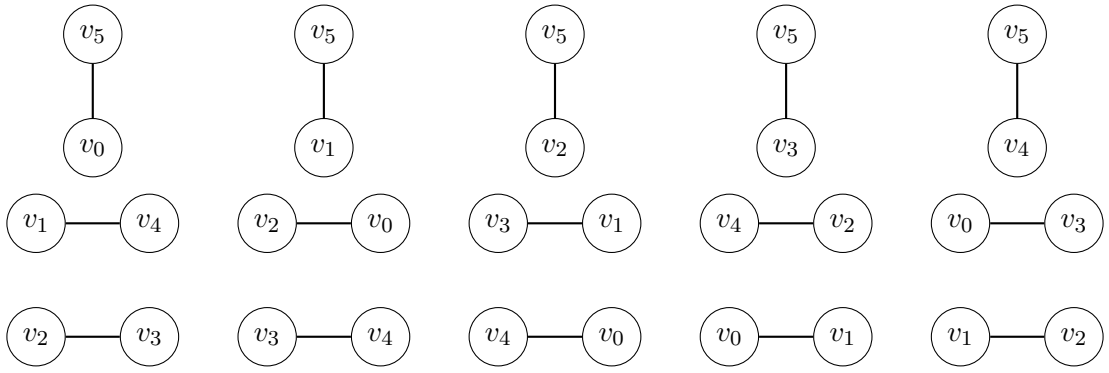
required matchings. It will be shown that there are $(f-1)$ and f disjoint maximum matchings for f even and f odd, respectively. This is done by giving construction rules for both cases. Let $\{v_0, \ldots, v_{f-1}\}$ be the vertex set of the K_f.

First, f even is considered. For $i \in \{0, \ldots, f-2\}$, the i-th matching contains the edges

$$\{\{v_{f-1}, v_i\}, \{v_{i-1 \bmod f-1}, v_{i+1 \bmod f-1}\}, \ldots, \{v_{i-\frac{f}{2}+1 \bmod f-1}, v_{i+\frac{f}{2}-1 \bmod f-1}\}\}.$$

An example of this construction for the K_6 can be found in Figure 6.

It remains to show that the matchings constructed by this rule are disjoint. The edge containing v_{f-1} is different for each matching, as v_{f-1} is adjacent to v_i in the i-th matching. For the other edges of the i-th matching, the construction rule states that v_j is adjacent to $v_{j-2(j-i) \bmod f-1}$. The vertex $v_{j-2(j-i) \bmod f-1}$ is equal to $v_{2i-j \bmod f-1}$. For each matching, the vertex $v_{2i-j \bmod f-1}$ is different, as $2i \bmod f-1$ has a different value for each $i \in \{0, \ldots, f-2\} \setminus \{j\}$ and $f-1$ odd.



(a) 1st matching. (b) 2nd matching. (c) 3rd matching. (d) 4th matching. (e) 5th matching.

Figure 6: Example for the construction of the 5 disjoint maximum matchings of the K_6.

The case f odd is handled the same way. For $i \in \{0, \ldots, f-1\}$, the i-th matching contains the edges

$$\{\{v_{i-1 \bmod f}, v_{i+1 \bmod f}\}, \ldots, \{v_{i-\lfloor \frac{f}{2} \rfloor \bmod f}, v_{i+\lfloor \frac{f}{2} \rfloor \bmod f}\}\}.$$

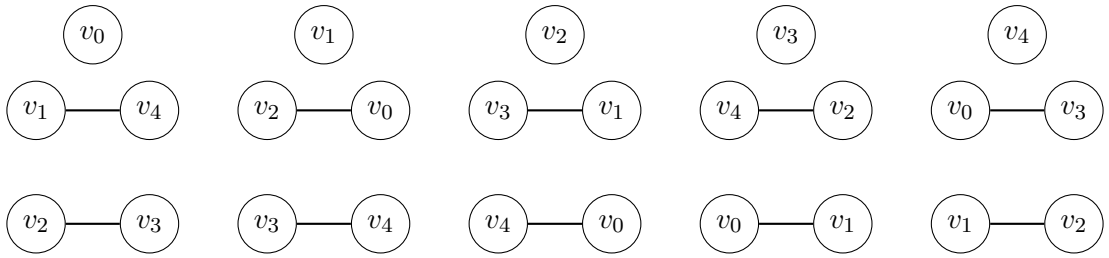
An example of this construction for the K_5 can be found in Figure 7.

Showing that the matchings constructed by this rule are disjoint is done with the same argument used for f even. For the i-th matching, the vertex v_j is adjacent to $v_{2i-j \bmod f}$ and these two vertices are only adjacent in the i-th matching, as $2i \bmod f$ has a different value for each $i \in \{0, \ldots, f-1\} \setminus \{j\}$ and f odd.

□

With Proposition 3.2 the lower bound on the number of $(K_3, \overline{R})$-free r-colorings of the $T_f(n)$ is

$$|\mathcal{C}_{r,(K_3,\overline{R})}(T_f(n))| > \begin{cases} \left(\frac{r}{f-1}\right)^{\frac{(f-1)n^2}{2f}} & \text{if } f \text{ is even} \\ \left(\frac{r}{f}\right)^{\frac{(f-1)n^2}{2f}} & \text{if } f \text{ is odd.} \end{cases}$$



(a) 1st matching. (b) 2nd matching. (c) 3rd matching. (d) 4th matching. (e) 5th matching.

Figure 7: Example for the construction of the 5 disjoint maximum matchings of the K_5.

Comparing the lower bounds on $|\mathcal{C}_{r,(K_3,\overline{R})}(T_f(n))|$ and $|\mathcal{C}_{r,(K_3,\overline{R})}(T_{f+1}(n))|$ for f odd gives

$$\left(\frac{r}{f}\right)^{\frac{(f-1)n^2}{2f}} \leq \left(\frac{r}{f}\right)^{\frac{fn^2}{2(f+1)}}$$
$$\left(\frac{r}{f}\right)^{\frac{f-1}{f}} \leq \left(\frac{r}{f}\right)^{\frac{f}{f+1}}.$$

Thus, the K_f for f even has a bigger lower bound and the case f odd is not considered further.

For f even, the upper bound $r_1(3,f)$ is calculated by

$$\begin{aligned}
|\mathcal{C}_{r,(K_3,\overline{R})}(T_2(n))| = r^{\frac{n^2}{4}} \leq \left(\frac{r}{f-1}\right)^{\frac{(f-1)n^2}{2f}} &< |\mathcal{C}_{r,(K_3,\overline{R})}(T_f(n))| \\
r^{\frac{1}{2}} &\leq \left(\frac{r}{f-1}\right)^{\frac{f-1}{f}} \\
r^{\frac{2-f}{2f}} &\leq \left(\frac{1}{f-1}\right)^{\frac{f-1}{f}} \\
r &\geq (f-1)^{\frac{2f^2-2f}{f^2-2f}} = (f-1)^{\frac{(f-1)2}{f-2}} =: r_1(3,f).
\end{aligned}$$

The upper bound $r_1(3)$ is determined by

$$r_1(3) = \min_{f>2, f \text{ even}} r_1(3,f).$$

The upper bound $r_1(k)$ for $3 \leq k \leq 8$ is stated in Table 1. These values of $r_1(k)$ were obtained by calculating $r_1(k,f)$ for $f = k, \ldots, \lceil e^k \rceil$ and taking the minimum of these values. For the minimum, the associated f cannot be larger than $\lceil e^k \rceil$ because the derivative of $r_1(k,f)$ with respect to f is nonnegative for $f \geq \lceil e^k \rceil$. For $f \geq \lceil e^k \rceil$ and $k = 3$, the derivative is

$$\begin{aligned}
r_1'(3,f) &= (f-1)^{\frac{2(f-1)}{f-2}} \cdot \left(\frac{2(f-2) + \ln(f-1) \cdot (2(f-2) - 2(f-1))}{(f-2)^2}\right) \\
&= (f-1)^{\frac{2(f-1)}{f-2}} \cdot \left(\frac{2(f-2) - 2\ln(f-1)}{(f-2)^2}\right) \\
&\geq 0.
\end{aligned}$$

For $k \geq 4$, the function

$$g(k,f) := \ln r_1(k,f) = \frac{(f-1)\cdot(k-1)}{f-k+1} \cdot \ln\left(\frac{(f-1)f}{2}\right)$$

is used, as the minimum of $g(k,f)$ and $r_1(k,f)$ is achieved for the same f. Then it holds that

$$\begin{aligned} g'(k,f) &= \frac{\ln\left(\frac{(f-1)f}{2}\right)(k-1)(f-k+1)f}{f(f-k+1)^2} + \frac{(2f-1)(k-1)(f-k+1)}{f(f-k+1)^2} \\ &\quad - \frac{\ln\left(\frac{(f-1)f}{2}\right)(k-1)(f-1)f}{f(f-k+1)^2} \\ &= \frac{k-1}{f(f-k+1)^2} \cdot \left(\frac{\ln\left(\frac{(f-1)f}{2}\right)f(2-k)}{f(f-k+1)^2} + \frac{(2f-1)(f-k+1)}{f(f-k+1)^2}\right) \\ &= \frac{k-1}{f(f-k+1)^2} \cdot \left(\frac{2\ln\left(\frac{(f-1)f}{2}\right)f - k\ln\left(\frac{(f-1)f}{2}\right)f + 2f^2 - 2fk + f + k - 1}{f(f-k+1)^2}\right) \\ &\geq 0, \end{aligned}$$

since $k \leq \ln((f-1)f/2)$ and $k\ln((f-1)f/2) \leq 2f$ for $f \geq \lceil e^k \rceil$. The choices of $f \geq \lceil e^k \rceil$ suffices for this thesis, but it should also be possible to argue that the derivative is nonnegative for smaller values of f.

k	3	4	5	6	7	8
$r_1(k)$	27	759375	$\approx 8.3038 \cdot 10^9$	$\approx 1.4071 \cdot 10^{14}$	$\approx 3.5567 \cdot 10^{18}$	$\approx 1.3087 \cdot 10^{23}$
f	4	6	10	15	19	24

Table 1: Upper bounds $r_1(k)$ for $3 \geq k \geq 8$ and the corresponding f that gives the minimum $r_1(k,f)$.

The value of $r_1(k)$ is only an upper bound on the r, where the $T_{k-1}(n)$ is no longer $(K_k, \overline{R})$-extremal and it is unknown if this bound is tight. Currently, there is no approach known to lower this upper bound, so $r_1(k)$ might be a good guess for the point at which the $(K_k, \overline{R})$-extremal graph changes.

From now on, the focus is on finding a lower bound on r such that the $T_{k-1}(n)$ is $(K_k, \overline{R})$-extremal. For an $r_0(k)$, it will be proven that for any $r \leq r_0(k)$ the $T_{k-1}(n)$ is the $(K_k, \overline{R})$-extremal graph. The meaning of $r_0(k)$ and $r_1(k)$ is illustrated in Figure 8.

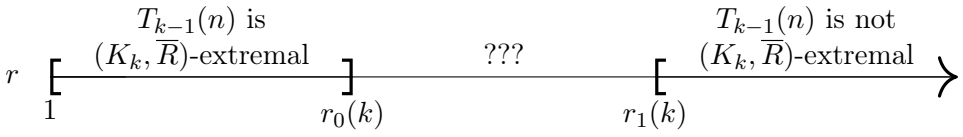


Figure 8: Illustration of what is known about the $(K_k, \overline{R})$-extremal property of the $T_{k-1}(n)$. For $r_0(k) < r < r_1(k)$, it is unknown if the $T_{k-1}(n)$ is $(K_k, \overline{R})$-extremal or not.

The lower bounds on $|\mathcal{C}_{r,(K_k,\overline{R})}(T_f(n))|$ that were presented in this section are also good lower bounds on the maximum number of distinct $(K_k, \overline{R})$-free r-colorings a graph on n ver-

tices can have. Moreover, the $T_f(n)$ that has the biggest lower bound on $|\mathcal{C}_{r,(K_k,\overline{R})}(T_f(n))|$ is a good candidate for a $(K_k, \overline{R})$-extremal graph for a given $r \geq r_1(k)$.

Before a value of $r_0(k)$ is given in the next section, the following theorem by Benevides, Hoppen and Sampaio [BHS17] is stated as an interesting result. From Theorem 3.4 it can be deduced that Turán graphs are good candidates as the $(K_k, \overline{R})$-extremal graphs, since they are complete multipartite and only the property of balanced partitions is different.

Theorem 3.4 (Theorem 1.1 from [BHS17])**.** *Let the pattern P be any pattern of the K_k. For every natural n, there exists a complete multipartite graph on n vertices that is (K_k, P)-extremal.*

3.3. Calculating the Lower Bound

In this section an approach is introduced to calculate the values of r where the $T_{k-1}(n)$ is known to be $(K_k, \overline{R})$-extremal. The approach used to prove the value of $\mathcal{C}_{r,(K_k,\overline{R})}(n)$ and the $(K_k, \overline{R})$-extremal graph is similar to [Alo+04; HLO15]. To state the corresponding theorem, the color stability property is required.

Definition 3.5 (Color Stability Property)**.** *Let F be a graph with chromatic number $\chi(F) = k \geq 3$ and let (F, P) be a color pattern. The color pattern (F, P) satisfies the color stability property for a positive integer r if for every $\delta > 0$ there exists an n_0 with the following property. If $n > n_0$ and G is a graph on n vertices such that $|\mathcal{C}_{r,(F,P)}(G)| \geq r^{\mathrm{ex}(n,F)}$, then there exists a partition $V(G) = V_1 \dot\cup \ldots \dot\cup V_{k-1}$ such that $\sum_{i=1}^{k-1} e(V_i) \leq \delta n^2$.*

The Color Stability Property is the main topic of Section 4 and is proven there for the non-rainbow pattern of the K_k. Furthermore, the definition of an $(r, k-1)$-vertex saturated pattern is required. This property of a color pattern (K_k, P) states that any arbitrary r-coloring of the graph $K_{1,k-1}$, which is a bipartite graph with one class of size 1 and the other class of size $(k-1)$, is part of a pattern in (K_k, P). The property is formally defined in Definition 3.6 and illustrated in Figure 9.

Definition 3.6 (Vertex saturated)**.** *For positive integers r and k, the color pattern (K_k, P) is called $(r, k-1)$-vertex saturated if for any r-coloring $C_{r,K_{1,k-1}}$ of the bipartite graph $K_{1,k-1}$, there exists an r-coloring $C_{r,K_k} \in (K_k, P)$ and a vertex $v \in V(K_k)$ such that the coloring of the edges incident with v for C_{r,K_k} is isomorphic to $C_{r,K_{1,k-1}}$.*

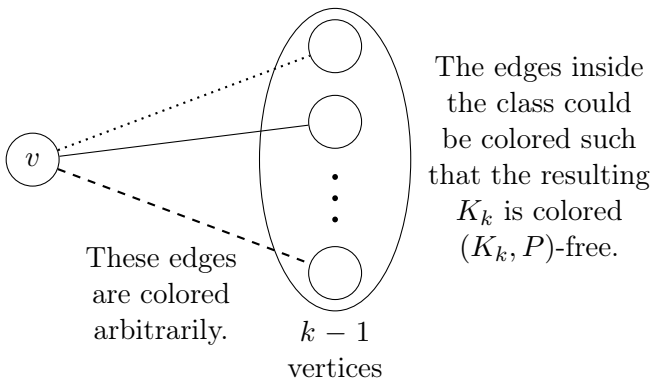


Figure 9: Illustration of Definition 3.6. If the edges between v and the partition are colored arbitrarily, then it is possible to color the resulting K_k (K_k, P)-free.

With these properties the main theorem can be stated.

Theorem 3.7. *For fixed integers $k \geq 3$ and $r \geq 2$, and let (K_k, P) be an $(r, k-1)$-vertex saturated color pattern. If (K_k, P) satisfies the Color Stability Property for r, then there exists an integer $n_0 > 0$ such that the following holds for $n > n_0$. If G is a graph on n vertices, then $|\mathcal{C}_{r,(K_k,P)}(G)| \leq r^{\mathrm{ex}(n,K_k)}$. Moreover, the only graph on n vertices for which the number of (K_k, P)-free r-colorings is $r^{\mathrm{ex}(n,K_k)}$ is the Turán graph $T_{k-1}(n)$.*

Combined with the Color Stability Property, Theorem 3.7 is a general theorem that proves the main result of this thesis for the color pattern $(K_k, P) = (K_k, \overline{R})$.

Proof of Theorem 3.7. For $k \geq 3$ and $r \geq 2$, and let (K_k, P) be an $(r, k-1)$-vertex saturated color pattern. Let (K_k, P) satisfy the Color Stability Property for r and n_0 is given by the Color Stability Property for $\delta = 1/(5^{2k+1} r^{4k} k^2)$.

Let $n > n_0$ and let G be a graph on n vertices such that $|\mathcal{C}_{r,(K_k,P)}(G)| \geq r^{\mathrm{ex}(n,K_k)}$. Assume for a contradiction that G is not isomorphic to $T_{k-1}(n)$. Since P is a pattern of K_k, Theorem 3.4 assures that there is one such G that is complete multipartite, and let $V(G) = V_1' \dot\cup \ldots \dot\cup V_s'$ be the associated partition of its vertex set. As G has at least $\mathrm{ex}(n, K_k)$ edges, it is $s \geq k$. Let $V(G) = V_1 \dot\cup \ldots \dot\cup V_{k-1}$ be a partition of the vertex set of G such that $\sum_{i=1}^{k-1} e(V_i)$ is minimal. The minimality of this partition implies that, for every $i, j \in [k-1]$ and every $v \in V_i$, it holds that $|N(v) \cap V_j| \geq |N(v) \cap V_i|$, otherwise moving v into V_j would give a partition with fewer edges inside the classes.

By the Color Stability Property, the sum $\sum_{i=1}^{k-1} e(V_i)$ is at most δn^2. Let G^* be the $(k-1)$-partite subgraph of G generated from the partition $V_1, \ldots, V_{k-1}$, so that $e(G^*) \geq e(G) - \delta n^2 \geq \mathrm{ex}(n, K_k) - \delta n^2$. By Lemma 2.21, for all $i \in [k-1]$,

$$|V_i| \geq \frac{n}{k-1} - \sqrt{2\delta} n.$$

Claim 3.8. *There exists a vertex v such that $|N(v) \cap V_i| \geq n/(2k)$ for all $i \in [k-1]$.*

Proof. Since G is not $(k-1)$-partite, there is an edge $\{u, v\}$ with both endpoints in one of the classes $V_1, \ldots, V_k$, say the class is V_1. This means that u and v are in different classes with respect to the partition $V_1' \dot\cup \ldots \dot\cup V_s'$. For every vertex $x \in V(G) \setminus \{u, v\}$, x must be adjacent to u or v, as G is complete multiparite. Assume w.l.o.g. that $|N(v) \cap V_1| \geq |N(u) \cap V_1|$ and it holds that

$$|N(v) \cap V_1| \geq \frac{|V_1|}{2} \geq \frac{n}{2(k-1)} - \sqrt{\delta} n \geq \frac{n}{2k},$$

as $\delta = 1/(5^{2k+1} r^{4k} k^2) \leq 1/(4k^2(k-1)^2)$. □

Using Claim 3.8, fix a vertex v, say $v \in V_1$, which is adjacent to at least $n/(2k)$ vertices in each V_i for $i \in [k-1]$.

Now the structure of a fixed (K_k, P)-free r-coloring C_{r,K_k} is analyzed. For $i \in [k-1]$, define $W_i = N(v) \cap V_i$. By pigeonhole principle, there are colors $\alpha_1, \ldots, \alpha_{k-1} \in [r]$, not necessarily distinct, and subsets $W_1^{\alpha_1}, \ldots W_{k-1}^{\alpha_{k-1}}$, where $W_i^{\alpha_i} \subseteq W_i$, such that all edges between v and $W_i^{\alpha_i}$ have color α_i and $|W_i^{\alpha_i}| \geq n/(2rk)$.

Since (K_k, P) is $(r, k-1)$-vertex saturated, there exists a vertex $x \in V(K_k)$ and a coloring $C_{r,K_k} \in (K_k, P)$ such that the colors of the $(k-1)$ edges incident with x in K_k are $\alpha_1, \ldots, \alpha_{k-1}$. Lemma 2.18 with $\omega(i) = 1/(5r^2)^i$ for every $i \in [k]$ and $\beta = 1/(5r^2)$ is applied to obtain a pair (X_i, X_j) and color $h \in [r]$ with the following properties. The pair (X_i, X_j) satisfies $X_i \subseteq W_i^{\alpha_i}$, where $|X_i| \geq |W_i^{\alpha_i}|/(5r^2)^{k-1}$ and $X_j \subseteq W_j^{\alpha_j}$, where $|X_j| \geq |W_j^{\alpha_j}|/(5r^2)^{k-1}$ and there are at most $|X_i||X_j|/(5r^2)$ edges of color h between $|X_i|$

and $|X_j|$. Otherwise, by Lemma 2.18, there are vertices $v_1, \dots, v_{k-1}$, where $v_i \in W_i^{\alpha_i}$, such that the colored subgraph of $C_{r,G}$ with the vertices $v, v_1, \dots, v_{k-1}$ and all edges between these vertices would be in the forbidden color pattern (K_k, P).

To estimate the size of $\mathcal{C}_{r,(K_k,P)}(G)$, the number of (K_k, P)-free r-colorings that may be associated with the pair (X_i, X_j) and color h is bounded. There are r choices for the color h and at most 2^{2n} choices for the sets X_i and X_j. By Lemma 2.23 and Lemma 2.24, for a fixed color h and fixed sets X_i and X_j, there are at most

$$\begin{aligned}\binom{|X_i||X_j|}{|X_i||X_j|/((5r^2)} (r-1)^{|X_i||X_j|} &\overset{(3)}{\leq} 2^{H(1/(5r^2))|X_i||X_j|}(r-1)^{|X_i||X_j|} \\ &\overset{(7)}{\leq} \left((5r^2)^{2/(5r^2)}(r-1)\right)^{|X_i||X_j|}\end{aligned}$$

ways to color the edges between X_i and X_j. So there are at most $\mathrm{ex}(n, K_k)+\delta n^2-|X_i||X_j|$ other edges in G. Thus, it holds that

$$\begin{aligned}|\mathcal{C}_{r,(K_k,P)}(G)| &\leq r2^{2n}(5r^2)^{(2/(5r^2))|X_i||X_j|}(r-1)^{|X_i||X_j|}r^{\mathrm{ex}(n,K_k)+\delta n^2-|X_i||X_j|} \\ &\leq \left(25r^4\left(\frac{r-1}{r}\right)^{5r^2}\right)^{|X_i||X_j|/(5r^2)} r^{\mathrm{ex}(n,K_k)+2\delta n^2}.\end{aligned}$$

Furthermore,

$$25r^4\left(\frac{r-1}{r}\right)^{5r^2} \leq 25r^4e^{-5r} < \frac{(2re^{-r})^5}{r} < \frac{1}{r}.$$

Finally, since $\delta = 1/(5^{2k+1}r^{4k}k^2)$ and by using the lower bounds on $|W_i^{\alpha_i}|$ and $|X_i|$,

$$|X_i||X_j| \geq \frac{n^2}{4r^2k^2(5r^2)^{2k-2}} > \frac{n^2}{5^{2k-1}r^{2(2k-1)}k^2},$$

it can be concluded that

$$\begin{aligned}|\mathcal{C}_{r,(K_k,P)}(G)| &< r^{\mathrm{ex}(n,K_k)+2\delta n^2-|X_i||X_j|/(5r^2)} \\ &< r^{\mathrm{ex}(n,K_k)+2\delta n^2-n^2/(5^{2k}r^{4k}k^2)} \\ &< r^{\mathrm{ex}(n,K_k)},\end{aligned}$$

which leads to the desired contradiction. □

Theorem 3.7 establishes that the $T_{k-1}(n)$ is (K_k, P)-extremal for a given r if the color pattern (K_k, P) is $(r, k-1)$-vertex saturated and satisfies the Color Stability Property for this r.

Considering the non-rainbow pattern $(K_k, \overline{R})$, it is trivial to show that this color pattern is $(r, k-1)$-vertex saturated for every $k \geq 3$ and $r \geq 2$. For any r-coloring of the $K_{1,k-1}$, it is possible to construct an r-coloring of the K_k that is non-rainbow by adding the missing edges to the $K_{1,k-1}$ and coloring these edges with the color of an edge $e \in E(K_{1,k-1})$.

The rest of this thesis is dedicated to show the Color Stability Property for the color pattern $(K_k, \overline{R})$.

4. The Color Stability Property

In this section the Color Stability Property, which is defined in Definition 3.5, is studied in detail for the color pattern $(K_k, \overline{R})$. The aim is to prove the Color Stability Property of $(K_k, \overline{R})$ for $k \geq 3$ and given values of r. The considered values of r, for which the Color Stability Property is proven, are given by Lemma 4.1. Together with Theorem 3.7, this proves that the $T_{k-1}(n)$ is $(K_k, \overline{R})$-extremal for $k \geq 3$ and $r \leq r_0(k)$.

Lemma 4.1 states this Color Stability Property for the color pattern $(K_k, \overline{R})$.

Lemma 4.1 (Color Stability Property for $(K_k, \overline{R})$)**.** *Fix an integer $k \geq 3$. There exists an integer $r_0(k) \geq \binom{k}{2}$ such that for all integers $2 \leq r \leq r_0(k)$ the following holds. For all $\delta > 0$, there exists n_0 such that if G is a graph on $n \geq n_0$ vertices that has at least $r^{\mathrm{ex}(n,K_k)}$ distinct $(K_k, \overline{R})$-free r-colorings, then there is a partition $V(G) = V_1 \dot\cup \ldots \dot\cup V_{k-1}$ such that*

$$\sum_{i=1}^{k-1} e(V_i) \leq \delta n^2.$$

Proof. The proof is divided into two cases. The first case is $2 \leq r < \binom{k}{2}$. As r is too small to color the K_k rainbow, graphs containing a K_k subgraph have no distinct $(K_k, \overline{R})$-free r-colorings. So only K_k-free graphs can have at least $r^{\mathrm{ex}(n,K_k)}$ r-colorings. Moreover, any r-coloring of a K_k-free graph is $(K_k, \overline{R})$-free. Thus, the number of distinct r-colorings of a K_f-free graph G' is $r^{e(G')}$. By Theorem 2.5, the only K_k-free graph on n vertices with at least $\mathrm{ex}(n, K_k)$ edges is the Turán graph $T_{k-1}(n)$. As the $T_{k-1}(n)$ is $(k-1)$-partite, there exists a partition $V(T_{k-1}(n)) = V_1 \dot\cup \ldots \dot\cup V_{k-1}$ such that

$$\sum_{i=1}^{k-1} e(V_i) = 0 \leq \delta n^2.$$

The second case is $\binom{k}{2} \leq r \leq r_0(k)$. Fix r such that $\binom{k}{2} \leq r \leq r_0(k)$ and let $\xi > 0$ and $\eta > 0$. Let $\varepsilon = \varepsilon(r, \eta, k) > 0$ as in Lemma 2.17 and assume w.l.o.g. that $\varepsilon < \eta/2$. Fix $M = M(r, \varepsilon)$ given by Lemma 2.15. Let G be a graph on $n \geq n_0$ vertices.

Let $C_{r,G}$ be an r-coloring of G that contains only rainbow colored K_k. By Lemma 2.15, there exists a multicolored ε-regular partition $V(G) = V_1 \dot\cup \ldots \dot\cup V_m$ of the colored graph, where $1/\varepsilon \leq m \leq M$.

At first, the number of distinct r-colorings of the edges of G omitted by the multicolored cluster graph are considered.

By Definition 2.12 and 2.14, there are at most $\varepsilon\binom{m}{2}$ irregular pairs for each color with respect to the partition $V(G) = V_1 \dot\cup \ldots \dot\cup V_m$. Thus, at most

$$r\varepsilon\binom{m}{2} \cdot \left(\frac{n}{m}\right)^2 \leq \frac{r\varepsilon}{2} \cdot n^2 \leq \frac{r\eta}{4} \cdot n^2 \tag{8}$$

edges of G are contained in an irregular pair with respect to some color. Moreover, using $m \geq 1/\varepsilon$ there are at most

$$m\left(\frac{n}{m}\right)^2 = \frac{n^2}{m} \leq \varepsilon n^2 \leq \frac{\eta}{2} \cdot n^2 \tag{9}$$

edges with both ends in the same V_i. Finally, for at most $\binom{m}{2}$ of the pairs $\{V_i, V_j\}$ and a fixed color, the edges-density $d(V_i, V_j)$ could be smaller than η, where the color could be

any of the r colors. Thus, for each of the r colors, the edges-density is at most η, which gives an upper bound on the total number of these edges of

$$r\eta\binom{m}{2}\cdot\left(\frac{n}{m}\right)^2 \leq \frac{r\eta}{2}\cdot n^2. \tag{10}$$

Using (8), (9) and (10) gives at most $r\eta n^2$ edges of these three types, which may be chosen in at most $\binom{n^2}{r\eta n^2}$ ways. This set of edges could be colored in at most $r^{r\eta n^2}$ different ways. By Lemma 2.23, the number of r-colorings of the edges of G that are omitted by a cluster graph are at most

$$r^{r\eta n^2}\cdot\binom{n^2}{r\eta n^2} \leq r^{r\eta n^2}\cdot 2^{H(r\eta)n^2}.$$

Now consider the edges represented in the multicolored cluster graph. Let $\mathcal{H}=\mathcal{H}(\eta)$ be the multicolored cluster graph associated with the partition $V(G)=V_1\dot\cup\ldots\dot\cup V_m$. For all $j\in[r]$, let $E_j(\mathcal{H})=\{e\in E(\mathcal{H}):|L_e|=j\}$ be the set of edges where the assigned list has size j and $e_j(\mathcal{H})=|E_j(\mathcal{H})|$. For the partition $V(G)=V_1\dot\cup\ldots\dot\cup V_m$ and the multicolored cluster graph $\mathcal{H}$, the number of r-colorings of G is bounded above by

$$r^{r\eta n^2}\cdot 2^{H(r\eta)n^2}\cdot\left(\prod_{j=1}^{r} j^{e_j(\mathcal{H})}\right)^{\left(\frac{n}{m}\right)^2}. \tag{11}$$

Let $\mathcal{H}^*$ be a multicolored cluster graph on m^* vertices that maximizes $\left(\prod_{j=1}^{r} j^{e_j(\mathcal{H})}\right)^{(n/m)^2}$. There are at most M^n possible partitions into $m\leq M$ classes and for each of these partitions there are at most $2^{rM^2/2}$ choices for the multicolored cluster graph. Thus, using (11), summing over all the partitions and all the corresponding multicolored cluster graphs, the number of r-colorings of G avoiding a non-rainbow colored K_k is bounded above by

$$\begin{aligned}
&M^n\cdot 2^{\frac{rM^2}{2}}\cdot 2^{H(r\eta)n^2}\cdot r^{r\eta n^2}\cdot\left(\prod_{j=1}^{r} j^{e_j(\mathcal{H}^*)}\right)^{\left(\frac{n}{m^*}\right)^2}\\
&\overset{n\gg 1}{\leq} 2^{\frac{3}{2}H(r\eta)n^2}\cdot r^{r\eta n^2}\cdot\left(\prod_{j=1}^{r} j^{e_j(\mathcal{H}^*)}\right)^{\left(\frac{n}{m^*}\right)^2}\\
&\overset{r\geq 3}{\leq} r^{(r\eta+H(r\eta))n^2}\cdot\left(\prod_{j=1}^{r} j^{e_j(\mathcal{H}^*)}\right)^{\left(\frac{n}{m^*}\right)^2}.
\end{aligned} \tag{12}$$

To continue the proof, the following claim is used. For simplicity, let $\ell=\ell(k)=\binom{k}{2}-1$, which is the maximum number of colors such that a K_k cannot be colored rainbow.

Claim 4.2. *There exists a multicolored cluster graph $\mathcal{H}$ such that*

$$e_{\lceil r/\ell\rceil}(\mathcal{H})+\ldots+e_r(\mathcal{H})\geq \mathrm{ex}(m,K_k)-\xi m^2.$$

Proof. Suppose for a contradiction that any r-coloring of G avoiding a non-rainbow colored K_k and any multicolored regular partition subject to this r-coloring induces a multicolored cluster graph $\mathcal{H}$ such that

$$e_{\lceil r/\ell\rceil}(\mathcal{H})+\ldots+e_r(\mathcal{H})< \mathrm{ex}(m,K_k)-\xi m^2. \tag{13}$$

First, suppose that $\binom{k}{2} \leq r \leq 2\ell$. Then $\lceil r/\ell \rceil \leq 2$ and the product $\left(\prod_{j=1}^{r} j^{e_j(\mathcal{H})}\right)^{(n/m)^2}$ can be estimated using only the assumption (13), since $j = 1$ has no impact on the product. By (12), (13) and choosing ξ such that

$$r^{(r\eta+H(r\eta))} < r^{\xi},$$

it holds for n large that

$$r^{(r\eta+H(r\eta))n^2} \cdot r^{\mathrm{ex}(n,K_k)-\xi n^2} \ll r^{\mathrm{ex}(n,K_k)},$$

which is a contradiction to the assumption of Lemma 4.1 that G has at least $r^{\mathrm{ex}(n,K_k)}$ $(K_k, \overline{R})$-free r-colorings.

Assume next that $r > 2\ell$. Given an ℓ-element subset $S \subset [r]$ and $j \in [r-\ell]$, let $E_j(S, int_{\geq 1}; \mathcal{H})$ be the set of all edges $e \in E_j(\mathcal{H})$ that satisfy $|L_e \cap S| \geq 1$ and let $e_j(S, int_{\geq 1}; \mathcal{H}) = |E_j(S, int_{\geq 1}; \mathcal{H})|$. This enables the following proposition.

Proposition 4.3. *Consider a $(K_k, \overline{R})$-free multicolored cluster graph $\mathcal{H}$.*

(a) *For every ℓ-element subset $S \subset [r]$ of colors, the subgraph $\mathcal{H}'$ of the multicolored cluster graph $\mathcal{H}$ with edge set*

$$\bigcup_{j=1}^{r-\ell} E_j(S, int_{\geq 1}; \mathcal{H}) \cup \bigcup_{p=r-\ell+1}^{r} E_p(\mathcal{H})$$

is K_k-free.

(b) *Moreover, there exists an ℓ-element subset $S \subset [r]$ such that*

$$\left|\bigcup_{j=1}^{r-\ell} E_j(S, int_{\geq 1}; \mathcal{H})\right| \geq \sum_{j=1}^{r-\ell} \frac{\binom{r}{\ell} - \binom{r-j}{\ell}}{\binom{r}{\ell}} \cdot |E_j(\mathcal{H})|.$$

Proof. Fix an ℓ-element subset $S \subset [r]$. First, part (a) will be proven. For a contradiction suppose that there is a K_k subgraph of $\mathcal{H}$ with edges $f_1, \ldots, f_{\ell+1}$. If at least one of these edges lies in $\bigcup_{p=r-\ell+1}^{r} E(\mathcal{H})$, then summing the sizes of the lists $L_{f_1}, \ldots, L_{f_{\ell+1}}$ gives at least $(r-\ell+1)+\ell = r+1$ because one of the lists has a size of at least $r-\ell+1$ and the other ℓ edges have a size of at least one. If even more edges lie in $\bigcup_{p=r-\ell+1}^{r} E(\mathcal{H})$, the lower bound on the sum of the list sizes can only increase. Therefore, at least one color appears in at least two lists, which produces a non-rainbow colored K_k, a contradiction.

Assume now that $f_1, \ldots, f_{\ell+1} \in \bigcup_{j=1}^{r-\ell} E_j(S, int_{\geq 1}; \mathcal{H})$. Summing the sizes of $L_{f_1} \cap S, \ldots, L_{f_{\ell+1}} \cap S$ gives at least $\ell+1$, as for each edge f_i the size of $L_{f_i} \cap S$ is at least one. So at least two of the lists contain the same color because S has size ℓ. This produces a non-rainbow colored K_k subgraph, a contradiction, which proves part (a).

For part (b), first it will be shown that

$$\sum_{S \in \binom{[r]}{\ell}} \sum_{j=1}^{r-\ell} |E_j(S, int_{\geq 1}; \mathcal{H})| = \sum_{j=1}^{r-\ell} \left(\binom{r}{\ell} - \binom{r-j}{\ell}\right) \cdot |E_j(\mathcal{H})|.$$

For $j \in [r-\ell]$, every edge $e \in E_j(\mathcal{H})$ is counted on the left hand side for all possible subsets $S \in \binom{[r]}{\ell}$ where $|L_e \cap S| \geq 1$. For each $j \in [r-\ell]$, these are $\binom{r}{\ell} - \binom{r-j}{\ell}$ many subsets, as

the number of possible subsets S is $\binom{r}{\ell}$ and there are $\binom{r-j}{\ell}$ many ℓ-element subsets that contain no element from a j-element subset.

By averaging, there exists an ℓ-element subset $S \subset [r]$ such that

$$\left|\bigcup_{j=1}^{r-\ell} E_j(S, int_{\geq 1}; \mathcal{H})\right| = \sum_{j=1}^{r-\ell} |E_j(S, int_{\geq 1}; \mathcal{H})| \geq \sum_{j=1}^{r-\ell} \frac{\binom{r}{\ell} - \binom{r-j}{\ell}}{\binom{r}{\ell}} \cdot |E_j(\mathcal{H})|,$$

as there are $\binom{r}{\ell}$ distinct choices for S and the sets $E_j(\mathcal{H})$ are pairwise disjoint. □

Proposition 4.3 leads to the inequality

$$\sum_{j=1}^{r-\ell} \frac{\binom{r}{\ell} - \binom{r-j}{\ell}}{\binom{r}{\ell}} \cdot e_j(\mathcal{H}) + \sum_{p=r-\ell+1}^{r} e_p(\mathcal{H}) \leq \mathrm{ex}(m, K_k). \tag{14}$$

Another inequality is obtained from the following proposition.

Proposition 4.4. *Consider a $(K_k, \overline{R})$-free multicolored cluster graph $\mathcal{H}$. For $q = 1, \ldots, \left\lfloor r/\binom{k}{2} \right\rfloor$, let $\mathcal{H}'_q$ be the subgraph of $\mathcal{H}$ with edge set $\bigcup_{j=q}^{r-\ell\cdot q} E_j(\mathcal{H})$ and fix a $(k-1)$-partite subgraph B'_q of $\mathcal{H}'_q$.*

(a) *The subgraph $\mathcal{H}''_q$ of the multicolored cluster graph $\mathcal{H}$ with edge set*

$$E(B'_q) \cup \bigcup_{p=r-\ell\cdot q+1}^{r} E_p(\mathcal{H})$$

is K_k-free.

(b) *Moreover, there exists a $(k-1)$-partite subgraph B'_q such that*

$$|E(B'_q)| \geq \frac{k-2}{k-1} \cdot |E(\mathcal{H}'_q)|.$$

Proof. First, part (a) will be proven. For a contradiction suppose that there is a K_k subgraph of $\mathcal{H}$ with edges $f_1, \ldots, f_{\ell+1}$. Since B'_q is $(k-1)$-partite, at least one of these edges lies in $\bigcup_{p=r-\ell\cdot q+1}^{r} E_p(\mathcal{H})$. Therefore, one of these edges has a list of size at least $r - \ell \cdot q + 1$ and the other ℓ edges have a list of size at least q. Summing the sizes of lists $L_{f_1}, \ldots, L_{f_{\ell+1}}$ gives at least $(r - \ell \cdot q + 1) + \ell \cdot q = r + 1$. If more of these edges lie in $\bigcup_{p=r-\ell\cdot q+1}^{r} E_p(\mathcal{H})$, then the lower bound on the sum of list sizes increases. As there are only r colors and the sum is at least $r + 1$, at least one color appears in at least two lists, which produces a non-rainbow colored K_k, a contradiction. This proves part (a).

For part (b) a well-known result about the maximum cut of a graph is used. For $k \geq 3$ and a graph G, G contains a $(k-1)$-partite subgraph with at least

$$\frac{k-2}{k-1} \cdot |E(G)|$$

edges. Thus, there exists a $(k-1)$-partite subgraph B'_q with

$$|E(B'_q)| \geq \frac{k-2}{k-1} \cdot |E(\mathcal{H}'_q)|.$$

□

For $q = 1, \ldots, \left\lfloor r/\binom{k}{2} \right\rfloor$, Proposition 4.4 leads to the inequality

$$\frac{k-2}{k-1} \cdot \sum_{j=q}^{r-\ell \cdot q} e_j(\mathcal{H}) + \sum_{p=r-\ell \cdot q+1}^{r} e_p(\mathcal{H}) \leq \operatorname{ex}(m, K_k). \tag{15}$$

As 1^x is 1, finding the maximum of $\prod_{j=1}^{r} j^{e_j(\mathcal{H}^*)}$ in (12) is equivalent to maximizing the sum

$$\sum_{j=2}^{r} e_j \cdot \ln j,$$

which is a linear function with respect to the variables $e_2, \ldots, e_r \geq 0$. Together with the linear constraints (14) and (15), a linear program could be formulated. Given a multicolored cluster graph $\mathcal{H}$, set

$$\zeta(\mathcal{H}) = (\operatorname{ex}(m, K_k) - e_{r-\ell+1}(\mathcal{H}) - \ldots - e_r(\mathcal{H}))/m^2 \tag{16}$$

such that $\zeta(\mathcal{H}) > \xi$ by (13), since $r - \ell \geq \lceil r/\ell \rceil$ for $r > 2\ell$ and $\ell \geq 2$. With (16) and $x_i = e_i(\mathcal{H})/(\zeta(\mathcal{H})m^2)$ the linear program is

$$\begin{aligned} \max \quad & x_2 \ln 2 + \ldots + x_{r-\ell} \ln (r-\ell) \\ \text{s.t.} \quad & \sum_{j=2}^{r-\ell} \frac{\binom{r}{\ell} - \binom{r-j}{\ell}}{\binom{r}{\ell}} \cdot x_j \leq 1 \\ & \frac{k-2}{k-1} \cdot \sum_{j=\max\{2,q\}}^{r-\ell \cdot q} x_j + \sum_{p=r-\ell \cdot q+1}^{r-\ell} x_p \leq 1, \quad q = 1, \ldots, \left\lfloor \frac{r}{\binom{k}{2}} \right\rfloor \\ & x_2, \ldots, x_{r-\ell} \geq 0. \end{aligned} \tag{17}$$

The source code to solve this linear program is given in Appendix A.2.

Let $y_k(r)$ be the optimal value of the linear program and $Y_k(r) = e^{y_k(r)}$. Furthermore, $r_0(k)$ is defined as the smallest positive integer r that satisfies the condition $Y_k(r+1) \geq r+1$. This implies that $Y_k(r) < r$ for all $2\ell < r \leq r_0(k)$.

Moreover, the value of $r_0(k)$ is at least $2\binom{k}{2}$. This will be shown by setting $r = 2\binom{k}{2}$ and using LP (17) with only the constraint resulting from Proposition 4.4 for $q = 2$. The resulting linear program is

$$\begin{aligned} \max \quad & x_2 \ln 2 + \ldots + x_{\binom{k}{2}+1} \ln \left(\binom{k}{2} + 1 \right) \\ \text{s.t.} \quad & \frac{k-2}{k-1} \cdot x_2 + \sum_{p=3}^{\binom{k}{2}+1} x_p \leq 1 \\ & x_2, \ldots, x_{\binom{k}{2}+1} \geq 0. \end{aligned} \tag{18}$$

Since adding more constraints from LP (17) to this linear program cannot increase $Y_k(r)$, the $Y_k(r)$ of LP (18) is an upper bound on the $Y_k(r)$ for LP (17). Thus, if $Y_k(r) < r$ for LP (18), then $Y_k(r) < r$ for LP (17). As the coefficients of $x_3, \ldots, x_{\binom{k}{2}+1}$ are 1, the best choice for theses variables is choosing $x_{\binom{k}{2}+1}$ as big as possible, which results in $Y_k(r) = \binom{k}{2} + 1$. Alternatively, a better choice could be to choose x_2 as big as possible,

which results in $Y_k(r) = 2^{(k-1)/(k-2)} \leq 4$ for $k \geq 3$. The maximum of these two possible $Y_k(r)$ is the $Y_k(r)$ of LP (18). Since 4 and $\binom{k}{2}+1$ are both smaller than $2\binom{k}{2}$, the $Y_k(r)$ of LP (18) is smaller than r for $r \leq 2\binom{k}{2}$.

For any multicolored cluster graph $\mathcal{H}$, it holds that

$$\begin{aligned}
\prod_{j=2}^{r} j^{e_j(\mathcal{H})} &= \left(\prod_{j=2}^{r-\ell} j^{e_j(\mathcal{H})}\right) \cdot \left(\prod_{j=r-\ell+1}^{r} j^{e_j(\mathcal{H})}\right) \\
&\leq \left(\prod_{j=2}^{r-\ell} j^{e_j(\mathcal{H})}\right) \cdot r^{e_{r-\ell+1}(\mathcal{H})+\ldots+e_r(\mathcal{H})} \\
&\overset{(16)}{=} \left(\prod_{j=2}^{r-\ell} j^{e_j(\mathcal{H})}\right) \cdot r^{\mathrm{ex}(m,K_k)-\zeta(\mathcal{H})m^2} \\
&\leq Y_k(r)^{\zeta(\mathcal{H})m^2} \cdot r^{\mathrm{ex}(m,K_k)-\zeta(\mathcal{H})m^2} \\
&< Y_k(r)^{\xi m^2} \cdot r^{\mathrm{ex}(m,K_k)-\xi m^2}, \qquad (19)
\end{aligned}$$

as $Y_k(r) < r$ and $\zeta(\mathcal{H}) > \xi$.

By (12), (19) and choosing ξ such that

$$r^{r\eta+H(r\eta)} < \left(\frac{r}{Y_k(r)}\right)^{\xi},$$

it holds for n large that

$$r^{(r\eta+H(r\eta))n^2} \cdot \left(\frac{Y_k(r)}{r}\right)^{\xi n^2} \cdot r^{\mathrm{ex}(n,K_k)} \overset{n\gg 1}{\ll} r^{\mathrm{ex}(n,K_k)}.$$

This implies that G has fewer than $r^{\mathrm{ex}(n,K_k)}$ distinct $(K_k, \overline{R})$-free r-colorings, a contradiction to the assumption of Lemma 4.1. This concludes the prove of Claim 4.2. □

Claim 4.2 guarantees that, for $\binom{k}{2} \leq r \leq r_0(k)$, there exists a cluster graph $\mathcal{H}$ with

$$e_{\lceil r/\ell \rceil}(\mathcal{H}) + \ldots + e_r(\mathcal{H}) \geq \mathrm{ex}(m, K_k) - \xi m^2. \qquad (20)$$

Let $\mathcal{H}'$ be a subgraph of $\mathcal{H}$ with edge set $E_{\lceil r/\ell \rceil} \cup \ldots \cup E_r$. By Theorem 2.19, there exists a partition $V(\mathcal{H}') = U_1 \dot\cup \ldots \dot\cup U_{k-1}$ with

$$e(U_1) + \ldots + e(U_{k-1}) \leq \xi m^2. \qquad (21)$$

Let $\widehat{\mathcal{H}}$ be the $(k-1)$-partite subgraph of $\mathcal{H}$ generated from the partition $U_1 \dot\cup \ldots \dot\cup U_{k-1}$ and with the maximum number of edges. By (20) and (21), it holds that

$$e(\widehat{\mathcal{H}}) \geq \mathrm{ex}(m, K_k) - 2\xi m^2.$$

By choosing

$$\xi < \frac{1}{8(k-1)^2},$$

it holds that

$$e_1(\mathcal{H}) + \ldots + e_{\lceil r/\ell \rceil - 1} \leq (4k-2)\xi m^2, \qquad (22)$$

as according to Lemma 2.20 the graph obtained by adding the edges $E_1 \cup \ldots \cup E_{\lceil r/\ell \rceil - 1}$ to $\widehat{\mathcal{H}}$ would contain a K_k subgraph such that one of the edges of the K_k, say f_1, is in $E_1 \cup \ldots \cup E_{\lceil r/\ell \rceil - 1}$. Let $f_2, \ldots, f_{\ell+1}$ be the other edges of the K_k, which lie in $E_{\lceil r/\ell \rceil} \cup \ldots \cup E_r$. Then summing the sizes of the lists $L_{f_1}, \ldots, L_{f_{\ell+1}}$ gives

$$|L_{f_1}| + \ldots + |L_{f_{\ell+1}}| \geq 1 + \ell \left\lceil \frac{r}{\ell} \right\rceil \geq 1 + r > r. \tag{23}$$

So at least one color appears in at least two lists. By Lemma 2.17, this generates a non-rainbow colored K_K in G, a contradiction.

Therefore, using (21) and (22), the number of edges of $\mathcal{H}$ with both endpoints in the same U_i is at most $(4k-1)\xi m^2$. Let $W_i = \bigcup_{j \in U_i} V_j$ for $i \in \{1, \ldots, k-1\}$. Then, by choosing η and ξ such that

$$\eta < \frac{\delta}{2r} \text{ and } \xi < \frac{\delta}{8k-2},$$

it holds that

$$\begin{aligned} e(W_1) + \ldots + e(W_{k-1}) &\leq r\eta n^2 + \left(\frac{n}{m}\right)^2 \cdot (e(U_1) + \ldots + e(U_{k-1})) \\ &\leq r\eta n^2 + \left(\frac{n}{m}\right)^2 \cdot (4k-1)\xi m^2 \\ &< \frac{r\delta n^2}{2r} + \frac{(4k-1)\delta n^2 m^2}{(8k-2)m^2} \\ &= \frac{\delta n^2}{2} + \frac{\delta n^2}{2} = \delta n^2, \end{aligned}$$

as required. This concludes the proof of Lemma 4.1. □

The lower bound on r such that the $T_{k-1}(n)$ is $(K_k, \overline{R})$-extremal is the $r_0(k)$ from Lemma 4.1 and is calculated by the linear program (17). For r small, the linear program could be solved with an algorithmic solver like *CPLEX*. As the $r_0(k)$ increases with increasing k, the $r_0(k)$ can be calculated by this method only for small k. In Table 2 the $r_0(k)$ for $k \leq 15$ calculated from the LP (17) is shown.

k	3	4	5	6	7	8	9	10	11	12	13	14	15
$r_0(k)$	12	74	182	346	576	882	1273	1757	2344	3041	3859	4806	5890

Table 2: Value of $r_0(k)$ for $3 \leq k \leq 15$ resulting from LP (17).

Comparing these lower bounds to the upper bounds from Table 1 shows that there is a big gap between these bounds and there is room for improvement. Nevertheless, the lower bound resulting from Lemma 4.1 shows that there are values of r where the $T_{k-1}(n)$ is $(K_k, \overline{R})$-extremal.

To improve the result of Lemma 4.1, the $r_0(k)$ has to be increased. The limiting factor for the value of $r_0(k)$ is the linear program (17). In the next section, the LP is studied more closely and some approaches are presented to improve $r_0(k)$.

5. Investigation of the Linear Program

The goal of this section is to modify LP (17) such that it increases the value of $r_0(k)$ in Lemma 4.1.

First, the objective function is investigated, as it cannot be changed. The coefficients of the objective function are monotonically increasing with increasing index. This means that the variables with large indexes increase the optimal value more than the variables with smaller indexes for the same value of the variable. So the variables with bigger indexes are preferred for the maximum value. Therefore, the higher the index, the more the variables need to be restricted to get a small optimal value.

In the next section, the notion of active constraints is introduced to evaluate which constraints restrict the optimal value and how the modifications improve the linear program. After that, some approaches to improve the LP are presented and evaluated. At the end of this section, possible constraints are discussed that could close the gap between $r_0(k)$ and $r_1(k)$.

5.1. The Active Constraints of a Linear Program

The active constraints of an optimization problem are the constraints that basically define the optimal solution of the problem. Conversely, if a constraint is not active, the constraint could be removed without changing the optimal solution. This is formally defined for a linear program in Definition 5.1.

Definition 5.1 (Active constraint of a LP)**.** *Let $a_i \in \mathbb{R}^{1\times n}$ be the the i-th row of the matrix $A \in \mathbb{R}^{m\times n}$ and b_i the i-th element of the vector $b \in \mathbb{R}^n$. The constraint $a_i x \leq b_i$ of the linear program*

$$\begin{aligned} \max \quad & c^T x \\ s.t. \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

is called active iff $a_i x^* = b_i$ for the optimal solution x^*.

In other words, the active constraints limit the value of the objective function and are the constraints that give the optimal value. This does not mean that the optimal value changes if one active constraint is removed, as by including the active constraint twice, removing one would not change the optimal value.

An approach to get the active constraints from the set of all constraints is derived directly from Definition 5.1. This can simply be done by checking if $a_i x^* = b_i$ for a given constraint and if this is true, this constraint is active.

The active constraints of LP (17) are now calculated to see which constraints are needed and which could be omitted. The active constraint property can only be calculated for a specific LP and not for all possible r. The LP that is considered in this section is the one for $r = r_0(k)$, as this LP limits $r_0(k)$ from increasing further. The first constraint is

$$\sum_{j=2}^{r-\ell} \frac{\binom{r}{\ell} - \binom{r-j}{\ell}}{\binom{r}{\ell}} \cdot x_j \leq 1. \tag{24}$$

The optimal solution and the value of the left hand side of (24) are stated in Table 3 for $k \leq 15$.

As the left hand side of (24) is always 1 in Table 3, the constraint (24) is always active in the LP (17) for $r = r_0(k)$.

For $q = 1, \ldots, \left\lfloor r/\binom{k}{2} \right\rfloor$, the other constraints are

$$\frac{k-2}{k-1} \cdot \sum_{j=\max\{2,q\}}^{r-\ell\cdot q} x_j + \sum_{p=r-\ell\cdot q+1}^{r-\ell} x_p \leq 1. \tag{25}$$

k	non-zero variables	left hand side of (24)
3	$x_3 = 1.25$, $x_4 = 0.75$	1.0
4	$x_3 \approx 0.4155$, $x_6 \approx 0.2534$, $x_{59} \approx 0.8310$	1.0
5	$x_3 \approx 0.3106$, $x_6 \approx 0.091$, $x_{155} \approx 0.9318$	1.0
6	$x_3 \approx 0.2403$, $x_6 \approx 0.0485$, $x_{304} \approx 0.9612$	1.0
7	$x_3 \approx 0.1949$, $x_6 \approx 0.0306$, $x_{516} \approx 0.9745$	1.0
8	$x_3 \approx 0.1635$, $x_7 \approx 0.0221$, $x_{801} \approx 0.9811$	1.0
9	$x_3 \approx 0.1408$, $x_7 \approx 0.0162$, $x_{1168} \approx 0.9858$	1.0
10	$x_3 \approx 0.1236$, $x_7 \approx 0.0125$, $x_{1625} \approx 0.9889$	1.0
11	$x_3 \approx 0.1101$, $x_8 \approx 0.0102$, $x_{2182} \approx 0.9908$	1.0
12	$x_3 \approx 0.0992$, $x_8 \approx 0.0083$, $x_{2846} \approx 0.9924$	1.0
13	$x_3 \approx 0.0903$, $x_8 \approx 0.0069$, $x_{3628} \approx 0.9937$	1.0
14	$x_3 \approx 0.0829$, $x_8 \approx 0.0058$, $x_{4536} \approx 0.9946$	1.0
15	$x_3 \approx 0.0766$, $x_9 \approx 0.0051$, $x_{5578} \approx 0.9953$	1.0

Table 3: Analysis of the constraint (24) for $3 \leq k \leq 15$ and $r = r_0(k)$. The non-zero variables of the optimal solution including their value and the left hand side of the constraint (24) is stated.

This this is not a single constraint, so the constraints for all possible q need to be considered. In Table 4 the constraints in (25) are analyzed.

k	non-zero variables	left hand side of (25)
3	$x_3 = 1.25$, $x_4 = 0.75$	1.0 for $q \leq 3$ and ≈ 0.75 for $q \geq 4$
4	$x_3 \approx 0.4155$, $x_6 \approx 0.2534$, $x_{59} \approx 0.8310$	1.0 for $q \leq 6$ and ≈ 0.8310 for $q \geq 7$
5	$x_3 \approx 0.3106$, $x_6 \approx 0.091$, $x_{155} \approx 0.9318$	1.0 for $q \leq 6$ and ≈ 0.9318 for $q \geq 7$
6	$x_3 \approx 0.2403$, $x_6 \approx 0.0485$, $x_{304} \approx 0.9612$	1.0 for $q \leq 6$ and ≈ 0.9612 for $q \geq 7$
7	$x_3 \approx 0.1949$, $x_6 \approx 0.0306$, $x_{516} \approx 0.9745$	1.0 for $q \leq 6$ and ≈ 0.9745 for $q \geq 7$
8	$x_3 \approx 0.1635$, $x_7 \approx 0.0221$, $x_{801} \approx 0.9811$	1.0 for $q \leq 7$ and ≈ 0.9811 for $q \geq 8$
9	$x_3 \approx 0.1408$, $x_7 \approx 0.0162$, $x_{1168} \approx 0.9858$	1.0 for $q \leq 7$ and ≈ 0.9858 for $q \geq 8$
10	$x_3 \approx 0.1236$, $x_7 \approx 0.0125$, $x_{1625} \approx 0.9889$	1.0 for $q \leq 7$ and ≈ 0.9889 for $q \geq 8$
11	$x_3 \approx 0.1101$, $x_8 \approx 0.0102$, $x_{2182} \approx 0.9908$	1.0 for $q \leq 8$ and ≈ 0.9908 for $q \geq 9$
12	$x_3 \approx 0.0992$, $x_8 \approx 0.0083$, $x_{2846} \approx 0.9924$	1.0 for $q \leq 8$ and ≈ 0.9924 for $q \geq 9$
13	$x_3 \approx 0.0903$, $x_8 \approx 0.0069$, $x_{3628} \approx 0.9937$	1.0 for $q \leq 8$ and ≈ 0.9937 for $q \geq 9$
14	$x_3 \approx 0.0829$, $x_8 \approx 0.0058$, $x_{4536} \approx 0.9946$	1.0 for $q \leq 8$ and ≈ 0.9946 for $q \geq 9$
15	$x_3 \approx 0.0766$, $x_9 \approx 0.0051$, $x_{5578} \approx 0.9953$	1.0 for $q \leq 9$ and ≈ 0.9953 for $q \geq 10$

Table 4: Analysis of the constraints in (25) for $3 \leq k \leq 15$, $r = r_0(k)$ and $q = 1, \ldots, \left\lfloor r/\binom{k}{2} \right\rfloor$. The non-zero variables of the optimal solution including their value and the left hand side of (25) with the corresponding value of q is stated.

From Table 4 it can be seen that (25) is only active for q small. The constraints from (25) can be divided into 3 groups. The first group is for q small enough such that q is at most the smallest index of the non-zero variables, which always seems to be 3. The second group has q such that q is bigger than the smallest index of the non-zero variables and at most the second smallest index of the non-zero variables. The third group is for q bigger than the second smallest index of the non-zero variables. The first two groups are the active constraints. Thus, the constraints in the third group could be removed.

The constraints from the first and second group are now analyzed in more detail. If the constraints from the second group and third group are removed from the LP (17), then $x_3 = 0$ and the other two non-zero variables are still non-zero. If the constraints from the first and the third group are removed, then x_3 is the only non-zero variable. Therefore, the first group is responsible for the non-zero variables with the two largest indexes and the second group is responsible for the non-zero variable with the smallest index.

This can also be seen by including the constraints step by step and solving the LP. If the LP contains only the constraint (24), then the only non-zero variable is x_3. By including the constraints from the first group, the variable x_3 gets more restricted and variables with higher indexes are now non-zero. Including the constraints of the second group restricts these non-zero variables more and the value of x_3 is non-zero again.

To lower the optimal value, a modifications to LP (24) should restrict the non-zero variables more. An alternative idea could be to remove variables from the LP by considering them outside the LP.

5.2. Modifications to the Linear Program

Now some ideas are presented to improve LP (17), where improving means reducing the optimal value. In Section 5.2.1 and 5.2.2 the constraints from LP (17) will be modified by generalizing the underlying propositions. This could also allow variables to be removed from the linear program, although the actual constraints might then become worse. After that, another idea to deduce constraints is presented and explored.

Before the ideas are introduced, a general statement about the constraints of a linear program is mentioned, which helps to investigate and understand the effects of the modifications.

Generally, if the coefficients of the variables in the constraints get larger, then the optimal value of the linear program gets smaller. This can be seen by Example 5.2 with one constraint and one variable. The only coefficient is $a \in \mathbb{R}$.

Example 5.2 (Simple LP)**.**

$$\begin{aligned} \max \quad & x \\ \text{s.t.} \quad & ax \leq 1 \\ & x \geq 0. \end{aligned}$$

The optimal value of Example 5.2 is $1/a$, as the only constraint is active for $x = 1/a$. So the larger a is, the smaller is the optimal value.

This observation not necessarily means that the coefficients for all variables need to be increased to lower the optimal value. Strictly speaking, it only means that if the coefficients of the non-zero variables are all unchanged or decreased, the optimal value will not decrease, as any linear program with a smaller optimal value will need to restrict the non-zero variables more, otherwise the non-zero variables could be chosen as before, which is valid, and the optimal value will remain the same. This is the basic idea behind increasing the coefficients to improve the linear program.

5.2.1. Generalizing Proposition 4.3

Proposition 4.3 is generalized by assuming S to be a y-element subset of $[r]$ with $1 \leq y \leq \ell$. This results in Proposition 5.3.

Proposition 5.3. *Let $y = 1, \dots, \ell$ be a positive integer and*

$$z(y) \coloneqq \min \left\{ z \in \mathbb{N} : \left(\binom{k}{2} - y \right) (z+1) + y > r \right\}, \tag{26}$$

thus

$$z(y) \coloneqq \left\lfloor \frac{r - \binom{k}{2}}{\binom{k}{2} - y} \right\rfloor + 1.$$

Consider a $(K_k, \overline{R})$-free multicolored cluster graph $\mathcal{H}$.

(a) *For every y-element subset $S \subset [r]$ of colors, the subgraph $\mathcal{H}'_y$ of the multicolored cluster graph $\mathcal{H}$ with edge set*

$$\bigcup_{j=1}^{z(y)} E_j(S, int_{\geq 1}; \mathcal{H}) \cup \bigcup_{p=z(y)+1}^{r} E_p(\mathcal{H})$$

is K_k-free.

(b) *Moreover, there exists a y-element subset $S \subset [r]$ such that*

$$\left| \bigcup_{j=1}^{z(y)} E_j(S, int_{\geq 1}; \mathcal{H}) \right| \geq \sum_{j=1}^{z(y)} \frac{\binom{r}{y} - \binom{r-j}{y}}{\binom{r}{y}} \cdot |E_j(\mathcal{H})|.$$

Proof. Fix a y-element subset $S \subset [r]$. First, part (a) will be proven. For a contradiction suppose that there is a K_k subgraph of $\mathcal{H}$ with edges $f_1, \dots, f_{\ell+1}$. If at least $\binom{k}{2} - y$ of these edges lie in $\bigcup_{p=z(y)+1}^{r} E(\mathcal{H})$, then summing the sizes of the lists $L_{f_1}, \dots, L_{f_{\ell+1}}$ gives at least

$$\left(\binom{k}{2} - y \right) (z(y) + 1) + y \tag{27}$$

distinct r-colorings because $\binom{k}{2} - y$ of the lists have a size of at least $z(y)+1$ and the other y edges have a size of at least one. By the choice of $z(y)$ in (26), the term (27) is bigger than r. If even more edges lie in $\bigcup_{p=z(y)+1}^{r} E(\mathcal{H})$, the lower bound on the sum of the list sizes can only increase. Therefore, at least one color appears in at least two lists, which produces a non-rainbow colored K_k, a contradiction.

Assume now that at least $y+1$ of the edges $f_1, \dots, f_{\ell+1}$ lie in $\bigcup_{j=1}^{z(y)} E_j(S, int_{\geq 1}; \mathcal{H})$. Summing the sizes of $L_{f_1} \cap S, \dots, L_{f_{\ell+1}} \cap S$ gives at least $y+1$, as for each of the at least $y+1$ edges $f_i \in \bigcup_{j=1}^{z(y)} E_j(S, int_{\geq 1}; \mathcal{H})$ the size of $L_{f_i} \cap S$ is at least one. So at least two of the lists contain the same color because S has size y. This produces a non-rainbow colored K_k, a contradiction, which proves part (a).

For part (b), first it will shown that

$$\sum_{S \in \binom{[r]}{y}} \sum_{j=1}^{z(y)} |E_j(S, int_{\geq 1}; \mathcal{H})| = \sum_{j=1}^{z(y)} \left(\binom{r}{y} - \binom{r-j}{y} \right) \cdot |E_j(\mathcal{H})|.$$

For $j \in [z(y)]$, every edge $e \in E_j(\mathcal{H})$ is counted on the left hand side for all possible subsets $S \in \binom{[r]}{y}$ where $|L_e \cap S| \geq 1$. For each $j \in [z(y)]$, these are $\binom{r}{y} - \binom{r-j}{y}$ many subsets, as

the number of possible subsets S is $\binom{r}{y}$ and there are $\binom{r-j}{y}$ many y-element subsets that contain no element from a j-element subset.

By averaging, there exists a y-element subset $S \subset [r]$ such that

$$\left|\bigcup_{j=1}^{z(y)} E_j(S, int_{\geq 1}; \mathcal{H})\right| = \sum_{j=1}^{z(y)} |E_j(S, int_{\geq 1}; \mathcal{H})| \geq \sum_{j=1}^{z(y)} \frac{\binom{r}{y} - \binom{r-j}{y}}{\binom{r}{y}} \cdot |E_j(\mathcal{H})|,$$

as there are $\binom{r}{y}$ distinct choices for S and the sets $E_j(\mathcal{H})$ are disjoint for $j \in [z(y)]$. □

For $y = 1, \ldots, \ell$, Proposition 5.3 leads to the inequality

$$\sum_{j=1}^{z(y)} \frac{\binom{r}{y} - \binom{r-j}{y}}{\binom{r}{y}} \cdot e_j(\mathcal{H}) + \sum_{p=z(y)+1}^{r} e_p(\mathcal{H}) \leq \operatorname{ex}(m, K_k). \tag{28}$$

The first attempt is to use this new constraint directly inside LP (17). This can be done without any problem, since $z(y) \leq r - \ell$ with equality for $y = \ell$. The modified LP with the constraints from (28) is

$$\begin{aligned}
\max \quad & x_2 \ln 2 + \cdots + x_{z(\ell)} \ln z(\ell) && (29)\\
\text{s.t.} \quad & \sum_{j=2}^{z(y)} \frac{\binom{r}{y} - \binom{r-j}{y}}{\binom{r}{y}} \cdot x_j + \sum_{p=z(y)+1}^{z(\ell)} x_p \leq 1, \quad y = 1, \ldots, \ell \\
& \frac{k-2}{k-1} \cdot \sum_{j=\max\{2,q\}}^{r-\ell\cdot q} x_j + \sum_{p=r-\ell\cdot q+1}^{z(\ell)} x_p \leq 1, \quad q = 1, \ldots, \left\lfloor \frac{r}{\binom{k}{2}} \right\rfloor \\
& x_2, \ldots, x_{z(\ell)} \geq 0.
\end{aligned}$$

The source code to solve the linear program is stated in Appendix A.3. Solving LP (29) gives the same $r_0(k)$ as with LP (17). An approach to explain for which y the constraint is active is to compare the coefficients for a fixed variable x_i and different y. Consider y such that $i \leq z(y)$. These x_i have a coefficient of

$$\frac{\binom{r}{y} - \binom{r-i}{y}}{\binom{r}{y}}. \tag{30}$$

This coefficient is equal to

$$\begin{aligned}
\frac{\binom{r}{y} - \binom{r-i}{y}}{\binom{r}{y}} &= 1 - \frac{\binom{r-i}{y}}{\binom{r}{y}} \\
&= 1 - \frac{(r-i)!}{y!(r-i-y)!} \cdot \frac{y!(r-y)!}{r!} \\
&= 1 - \frac{(r-i)!}{r!} \cdot \frac{(r-y)!}{(r-i-y)!} \\
&= 1 - \frac{(r-i)!}{r!} \cdot (r-y)^{\underline{i}},
\end{aligned}$$

where $x^{\underline{i}} = x \cdot (x-1) \cdot \ldots \cdot (x-i+1)$ is the falling factorial. The falling factorial is defined for $x \geq i$. This condition is fulfilled as $(r-y) \geq (r-\ell) \geq i$. As $(r-y)^{\underline{i}}$ becomes smaller

with increasing y, the coefficient gets bigger with increasing y. Therefore, it is best to choose y as large as possible to obtain a small optimal value.

The next approach uses y outside the linear program, so that it is a parameter of the linear program. Then all variables x_i with $i \geq z(y)+1$, which have a coefficient of 1, can be removed from the LP by using

$$\zeta(\mathcal{H}) = (\mathrm{ex}(m, K_k) - e_{z(y)+1}(\mathcal{H}) - \ldots - e_r(\mathcal{H}))/m^2. \tag{31}$$

Since $z(y)+1$ is at least $\lceil r/\ell \rceil$, it holds that $\zeta(\mathcal{H}) > \xi$ by (13). The estimation (19) is adjusted by using $z(y)$ instead of $r-\ell$ as the bound of the product. The estimation is now

$$\begin{aligned}
\prod_{j=2}^{r} j^{e_j(\mathcal{H})} &= \left(\prod_{j=2}^{z(y)} j^{e_j(\mathcal{H})}\right) \cdot \left(\prod_{j=z(y)+1}^{r} j^{e_j(\mathcal{H})}\right) \\
&\leq \left(\prod_{j=2}^{z(y)} j^{e_j(\mathcal{H})}\right) \cdot r^{e_{z(y)+1}(\mathcal{H})+\ldots+e_r(\mathcal{H})} \\
&\overset{(31)}{=} \left(\prod_{j=2}^{z(y)} j^{e_j(\mathcal{H})}\right) \cdot r^{\mathrm{ex}(m,K_k)-\zeta(\mathcal{H})m^2} \\
&< Y_k(r)^{\zeta(\mathcal{H})m^2} \cdot r^{\mathrm{ex}(m,K_k)-\zeta(\mathcal{H})m^2} \\
&< Y_k(r)^{\xi m^2} \cdot r^{\mathrm{ex}(m,K_k)-\xi m^2},
\end{aligned}$$

as $Y_k(r) < r$ and $\zeta(\mathcal{H}) > \xi$ by (13).

Furthermore, inequality (15) cannot be used for all $q = 1, \ldots, \left\lfloor r/\binom{k}{2} \right\rfloor$, as for some q the coefficients of some removed variables are smaller than 1. The problem with coefficients smaller than 1 is that the constraint is more restrictive than inequality (15). This can be seen by Example 5.4.

Example 5.4. Consider the inequality

$$a \cdot e_2(\mathcal{H}) + b \cdot e_3(\mathcal{H}) + e_4(\mathcal{H}) \leq \mathrm{ex}(m, K_k),$$

where e_3 and e_4 should be outside the LP and $b < 1$. Using $\zeta(\mathcal{H}) = (\mathrm{ex}(m, K_k) - e_3(\mathcal{H}) - e_4(\mathcal{H}))/m^2$ and $x_i = e_i(\mathcal{H})/(\zeta(\mathcal{H})m^2)$, the inequality is transformed to

$$\begin{aligned}
a \cdot e_2(\mathcal{H}) + b \cdot e_3(\mathcal{H}) + e_4(\mathcal{H}) &\leq \mathrm{ex}(m, K_k) \\
a \cdot x_2 \cdot (\mathrm{ex}(m, K_k) - e_3(\mathcal{H}) - e_4(\mathcal{H})) &\leq \mathrm{ex}(m, K_k) - b \cdot e_3(\mathcal{H}) - e_4(\mathcal{H}) \\
a \cdot x_2 &\leq \frac{\mathrm{ex}(m, K_k) - b \cdot e_3(\mathcal{H}) - e_4(\mathcal{H})}{\mathrm{ex}(m, K_k) - e_3(\mathcal{H}) - e_4(\mathcal{H})} \\
a \cdot x_2 &\leq 1 + \frac{1-b}{\mathrm{ex}(m, K_k) - e_3(\mathcal{H}) - e_4(\mathcal{H})}.
\end{aligned}$$

This does not eliminate the variables $e_3(\mathcal{H})$ and $e_4(\mathcal{H})$ and estimating the right hand side with

$$a \cdot x_2 \leq 1 \leq 1 + \frac{1-b}{\mathrm{ex}(m, K_k) - e_3(\mathcal{H}) - e_4(\mathcal{H})}$$

reduces the maximum value that could be obtained with this constraint. This constraint is then invalid, as the resulting $r_0(k)$ is bigger than actually possible with the given inequality.

Therefore, q needs to be restricted to $q = \lceil (r - z(y))/\ell \rceil, \ldots, \lfloor r/k \rfloor$. For $y = 1, \ldots, \ell$, this leads to the linear program

$$\begin{aligned} \max \quad & x_2 \ln 2 + \cdots + x_{z(y)} \ln z(y) \\ \text{s.t.} \quad & \sum_{j=2}^{z(y')} \frac{\binom{r}{y'} - \binom{r-j}{y'}}{\binom{r}{y'}} \cdot x_j + \sum_{p=z(y')+1}^{z(y)} x_p \leq 1, \quad y' = 1, \ldots, y \\ & \frac{k-2}{k-1} \cdot \sum_{j=\max\{2,q\}}^{r-\ell \cdot q} x_j + \sum_{p=r-\ell \cdot q+1}^{z(y)} x_p \leq 1, \quad q = \left\lceil \frac{r - z(y)}{\ell} \right\rceil, \ldots, \left\lfloor \frac{r}{\binom{k}{2}} \right\rfloor \\ & x_2, \ldots, x_{z(y)} \geq 0. \end{aligned} \tag{32}$$

In Appendix A.4 the source code is stated to solve LP 32. Solving LP (32) for all possible values of y gives no improvement and even bigger optimal values, thus $r_0(k)$ does not increase. This results can be seen by considering the non-zero variables of the optimal solution with the smallest and the biggest index. Based on the known solutions of LP (17), if the smallest non-zero variable is x_d, the biggest non-zero variable is $x_{r-d\ell}$. If $x_{r-d\ell}$ is removed from the LP, then q is at least $d + 1$, as $x_{r-d\ell}$ in the constraint for $q = d$ has a coefficient with a value smaller than 1, which is removed from the LP.

Also, it was previously shown that the coefficient of the variables becomes smaller if the y gets smaller. Therefore, removing the constraint for $q = d$ such that x_d is the non-zero variable with the smallest index gives that x_d is restricted only by (28), so the optimal value can only increase.

In summary, this approach alone does not bring any improvement. The best choice for y seems to be ℓ and is known to be ℓ for $3 \leq k \leq 15$.

5.2.2. Generalizing Proposition 4.4

Proposition 4.4 is generalized by considering s-partite subgraphs instead of $(k-1)$-partite subgraphs, where $s < k$. Proposition 5.5 is the modified proposition. For the following calculations, the exact Turán number is used as defined in (1).

Proposition 5.5. *Let $s = 1, \ldots, k-1$, $q = 1, \ldots, \left\lfloor r/\binom{k}{2} \right\rfloor$ and*

$$t(s,q) := \left\lfloor \frac{r - \operatorname{ex}(k, K_{s+1}) \cdot q}{\binom{k}{2} - \operatorname{ex}(k, K_{s+1})} \right\rfloor.$$

Consider a $(K_k, \overline{R})$-free multicolored cluster graph $\mathcal{H}$. Let $\mathcal{H}'_{s,q}$ be the subgraph of $\mathcal{H}$ with edge set $\bigcup_{j=q}^{t(s,q)} E_j(\mathcal{H})$ and fix an s-partite subgraph $B'_{s,q}$ of $\mathcal{H}'_{s,q}$.

(a) *The subgraph $\mathcal{H}''_{s,q}$ of the multicolored cluster graph $\mathcal{H}$ with edge set*

$$E(B'_{s,q}) \cup \bigcup_{p=t(s,q)+1}^{r} E_p(\mathcal{H})$$

is K_k-free.

(b) *Moreover, there exists an s-partite subgraph $B'_{s,q}$ such that*

$$|E(B'_{s,q})| \geq \frac{s-1}{s} \cdot |E(\mathcal{H}'_{s,q})|.$$

Proof. First, part (a) will be proven. For a contradiction suppose that there is a K_k subgraph of $\mathcal{H}$ with edges $f_1, \dots, f_{\ell+1}$. Since $B'_{s,q}$ is s-partite, at least $\binom{k}{2} - \operatorname{ex}(k, K_{s+1})$ of these edges lie in $\bigcup_{p=t(s,q)+1}^{r} E_p(\mathcal{H})$. Therefore, $\binom{k}{2} - \operatorname{ex}(k, K_{s+1})$ of the edges have a list of size at least $t(s,q)+1$ and the other $\operatorname{ex}(k, K_{s+1})$ edges have a list of size at least q. Summing the sizes of lists $L_{f_1}, \dots, L_{f_{\ell+1}}$ gives at least

$$\begin{aligned}
&\operatorname{ex}(k, K_{s+1}) \cdot q + \left(\binom{k}{2} - \operatorname{ex}(k, K_{s+1}) \right) (t(s,q)+1) \\
&\quad = \operatorname{ex}(k, K_{s+1}) \cdot q + \left(\binom{k}{2} - \operatorname{ex}(k, K_{s+1}) \right) \left(\left\lfloor \frac{r - \operatorname{ex}(k, K_{s+1}) \cdot q}{\binom{k}{2} - \operatorname{ex}(k, K_{s+1})} \right\rfloor + 1 \right) \\
&\quad > \operatorname{ex}(k, K_{s+1}) \cdot q + \left(\binom{k}{2} - \operatorname{ex}(k, K_{s+1}) \right) \frac{r - \operatorname{ex}(k, K_{s+1}) \cdot q}{\binom{k}{2} - \operatorname{ex}(k, K_{s+1})} \\
&\quad = \operatorname{ex}(k, K_{s+1}) \cdot q + r - \operatorname{ex}(k, K_{s+1}) \cdot q \\
&\quad = r.
\end{aligned}$$

If more edges lie in $\bigcup_{p=t(s,q)+1}^{r} E_p(\mathcal{H})$, then the lower bound on the sum increases. As there are only r colors and the sum is bigger than r, at least one color appears in at least two lists, which produces a non-rainbow colored K_k, a contradiction. This proves part (a).

For part (b) a well-known result about the maximum cut of a graph is used. For $s \geq 1$, the graph G contains an s-partite subgraph with at least

$$\frac{s-1}{s} \cdot |E(G)|$$

edges. Thus, there exists an s-partite subgraph $B'_{s,q}$ with

$$|E(B'_{s,q})| \geq \frac{s-1}{s} \cdot |E(\mathcal{H}'_{s,q})|.$$

□

For $q = 1, \dots, \left\lfloor r / \binom{k}{2} \right\rfloor$ and $s = 1, \dots, k-1$, Proposition 5.5 leads to the inequality

$$\frac{s-1}{s} \cdot \sum_{j=q}^{t(s,q)} e_j(\mathcal{H}) + \sum_{p=t(s,q)+1}^{r} e_p(\mathcal{H}) \leq \operatorname{ex}(m, K_k). \tag{33}$$

As with Proposition 5.3, the first attempt is to use the constraint inside LP (17). This can be done without any problems, as $t(s,q) \leq r - \ell$ with equality for $s = k-1$ and $q = 1$. The linear program with the constraints from inequality (33) is

$$\begin{aligned}
\max \quad & x_2 \ln 2 + \dots + x_{t(k-1,1)} \ln t(k-1,1) && (34) \\
\text{s.t.} \quad & \sum_{j=2}^{t(k-1,1)} \frac{\binom{r}{\ell} - \binom{r-j}{\ell}}{\binom{r}{\ell}} \cdot x_j \leq 1 \\
& \frac{s-1}{s} \cdot \sum_{j=\max\{2,q\}}^{t(s,q)} x_j + \sum_{p=t(s,q)+1}^{t(k-1,1)} x_p \leq 1, \quad q = 1, \dots, \left\lfloor \frac{r}{\binom{k}{2}} \right\rfloor \text{ and } s = 1, \dots, k-1 \\
& x_2, \dots, x_{t(k-1,1)} \geq 0.
\end{aligned}$$

The source code to solve LP (34) is given in Appendix A.5. For $k \geq 4$, the optimal value of LP (34) is smaller than the optimal solution of LP (17), thus the $r_0(k)$ is larger.

Therefore, LP (34) is better than LP (17). Furthermore, the results of LP (34) are the new reference values to be improved upon. The $r_0(k)$ for $k \leq 7$ calculated from LP (34) is shown in Table 5.

k	3	4	5	6	7
$r_0(k)$	12	91	617	4694	48164

Table 5: Value of $r_0(k)$ for $3 \leq k \leq 7$ resulting from LP (34).

Since the value of $r_0(k)$ improves for $k \geq 4$, the active constraints are analyzed in detail. The optimal solution of LP (34) for $r = r_0(k)$ is given in Table 6.

k	non-zero variables
3	$x_3 = 1.25, x_4 = 0.75$
4	$x_4 = 0.25, x_{10} \approx 0.1001, x_{11} \approx 0.4496, x_{37} \approx 0.2003, x_{71} = 0.5$
5	$x_9 \approx 0.1111, x_{16} \approx 0.0375, x_{17} \approx 0.018, x_{38} \approx 0.0185, x_{41} \approx 0.037, x_{44} \approx 0.0188,$ $x_{47} \approx 0.2771, x_{130} \approx 0.0376, x_{140} \approx 0.1111, x_{240} \approx 0.0361, x_{244} \approx 0.0751,$ $x_{272} \approx 0.2222, x_{536} \approx 0.3333$
6	$x_{28} = 0.0625, x_{46} \approx 0.0208, x_{84} \approx 0.0417, x_{148} \approx 0.0139, x_{172} \approx 0.0278,$ $x_{227} \approx 0.0208, x_{234} \approx 0.0319, x_{235} \approx 0.1556, x_{656} \approx 0.0417, x_{713} \approx 0.0208,$ $x_{740} = 0.0625, x_{1228} \approx 0.0833, x_{1380} \approx 0.0417, x_{1452} = 0.125, x_{2048} = 0.0625,$ $x_{2165} = 0.1875, x_{4302} = 0.25$
7	$x_{148} = 0.04, x_{218} = 0.01, x_{349} \approx 0.0167, x_{669} \approx 0.0133, x_{674} = 0.0075,$ $x_{684} = 0.0125, x_{1256} \approx 0.0111, x_{1399} \approx 0.0089, x_{1429} = 0.005, x_{1435} \approx 0.0083,$ $x_{1616} \approx 0.0067, x_{1619} = 0.00375, x_{1623} = 0.00625, x_{1796} \approx 0.0389, x_{1805} \approx 0.1111,$ $x_{4439} = 0.0125, x_{4452} = 0.0075, x_{4459} \approx 0.0133, x_{4886} \approx 0.0167, x_{5060} = 0.01,$ $x_{5154} = 0.04, x_{7444} = 0.025, x_{7476} = 0.015, x_{7492} \approx 0.0267, x_{8516} \approx 0.0333,$ $x_{8935} = 0.02, x_{9159} = 0.08, x_{13960} = 0.05, x_{14746} = 0.03, x_{15166} = 0.12,$ $x_{22011} = 0.04, x_{22676} = 0.16, x_{45204} = 0.2$

Table 6: Non-zero variables of the optimal solution of LP (34) for $r = r_0(k)$ and $3 \leq k \leq 7$.

The first constraint, which is obtained from Proposition 4.3, is active for all $3 \leq k \leq 7$, as the left hand side of the constraint is 1 for this optimal solution. The constraints obtained from Proposition 5.5 are active for $s = 2, \ldots, k-1$ but not necessarily for all possible q. The constraints for $s = 1$ are not active. Thus, all the constraints for $s = 1$ could be removed. Removing the constraints for $s = 2, \ldots, k-1$, which are not active, is not really possible, as finding the values of q, for which the constraint are not active, cannot be done before solving the linear program. At least no approach is known to calculate these q in advance.

An observation about the indexes of the non-zero variables is that for some s and some non-zero variables x_q, the variable $x_{t(s,q)}$ is also non-zero. An example of this for $k = 5$ can be found in Example 5.6. This behavior could be explained by (33), as the value of x_q is already given by the remaining linear program and then $x_{t(s,q)}$, which is the biggest index in the first sum of (33), is chosen as big as possible.

Example 5.6. Let $k = 5$ and $r = 617$. The following non-zero variables of the optimal solution of LP (34) are related by $t(s, q)$.

$$x_9 \to x_{536}, \text{ for } s = 4$$
$$x_9 \to x_{272}, \text{ for } s = 3$$

$$x_9 \to x_{140}, \text{ for } s = 2$$
$$x_{16} \to x_{244}, \text{ for } s = 3$$
$$x_{16} \to x_{130}, \text{ for } s = 2$$

As with Proposition (5.3), the next attempt is to use s and q as a parameter outside the linear program. However, using q outside the LP is a bad idea and will stay inside the LP, as $t(s,q)$ can only be decreased if the lower bound on q increases for s fixed. The lower bound on q needs to be small because if x_3 is not restricted by (33), then x_3 is only restricted by (14), which will increase the optimal value. Furthermore, the effect of increasing the lower bound on q compared to decreasing the upper bound on s is relatively small. Nevertheless, this attempt cannot work, since the coefficient of $e_{t(s,1)}$ in inequality (14) for $s \leq k-2$ is smaller than 1 and $x_{t(s,1)}$ cannot be removed.

This problem can be solved by using the generalized version of Proposition 4.3. Therefore, the next attempt is to use the constraints from Proposition 5.3 and Proposition 5.5 in one LP. First, consider y as a parameter outside the linear program. For $y = 1, \ldots, \ell$ and using (31), the resulting linear program is

$$\begin{aligned} \max \quad & x_2 \ln 2 + \cdots + x_{z(y)} \ln z(y) && (35)\\ \text{s.t.} \quad & \sum_{j=2}^{z(y')} \frac{\binom{r}{y'} - \binom{r-j}{y'}}{\binom{r}{y'}} \cdot x_j + \sum_{p=z(y')+1}^{z(y)} x_p \leq 1, \quad y' = 1, \ldots, y \\ & \frac{s-1}{s} \cdot \sum_{j=\max\{2,q\}}^{t(s,q)} x_j + \sum_{p=t(s,q)+1}^{z(y)} x_p \leq 1, \quad (s,q) \in A(y) \\ & x_2, \ldots, x_{z(y)} \geq 0, \end{aligned}$$

where

$$A(y) = \left\{ \left(s \in \{1, \ldots, k-1\}, q \in \left\{ 1, \ldots, \left\lfloor r / \binom{k}{2} \right\rfloor \right\} \right) : t(s,q) \leq z(y) \right\}.$$

In Appendix A.6 the source code is stated to solve this linear program. The optimal solution of LP (35) is the same as LP (34). Moreover, the optimal solution is achieved for $y = \ell$. So this attempt does not bring any improvement.

In the previous attempt the value of y was chosen first and then s and q were chosen such that $t(s,q) \leq z(y)$. The next attempt is to choose s first and outside the linear program and then y is chosen such that $z(y) \leq t(s,1)$, as the maximum of $t(s,q)$ is achieved for $q = 1$. Removing the variables x_i with $i \geq t(s,1) + 1$ requires

$$\zeta(\mathcal{H}) = (\mathrm{ex}(m, K_k) - e_{t(s,1)+1}(\mathcal{H}) - \ldots - e_r(\mathcal{H}))/m^2. \tag{36}$$

Since $t(s,1) + 1$ is at least $\lceil r/\ell \rceil$, it holds that $\zeta(\mathcal{H}) > \xi$ by (13). This requires $s = 2, \ldots, k-1$, as $t(1,1)+1$ could be smaller than $\lceil r/\ell \rceil$. However, this is not a big problem, as the coefficients of the variables with index at most $t(s,q)$ are zero in the constraints for $s = 1$. So these constraints do not appear to be very restrictive when they are considered on their own.

For $s = 2, \dots, k-1$ and using (36), this results in the linear program

$$\begin{aligned} \max \quad & x_2 \ln 2 + \dots + x_{t(s,1)} \ln t(s,1) \\ \text{s.t.} \quad & \sum_{j=2}^{z(y)} \frac{\binom{r}{y} - \binom{r-j}{y}}{\binom{r}{y}} \cdot x_j + \sum_{p=z(y)+1}^{t(s,1)} x_p \leq 1, \quad y = 1, \dots, \operatorname{ex}(k, K_{s+1}) \\ & \frac{s'-1}{s'} \cdot \sum_{j=\max\{2,q\}}^{t(s',q)} x_j + \sum_{p=t(s',q)+1}^{t(s,1)} x_p \leq 1, \quad q = 1, \dots, \left\lfloor \frac{r}{\binom{k}{2}} \right\rfloor \text{ and } s' = 1, \dots, s \\ & x_2, \dots, x_{t(s,1)} \geq 0. \end{aligned} \tag{37}$$

The upper bound on y for the first constraint can be seen by calculating

$$\begin{aligned} t(s,1) &= \left\lfloor \frac{r - \operatorname{ex}(k, K_{s+1})}{\binom{k}{2} - \operatorname{ex}(k, K_{s+1})} \right\rfloor \\ &= \left\lfloor \frac{r - \operatorname{ex}(k, K_{s+1}) + \operatorname{ex}(k, K_{s+1}) - \binom{k}{2}}{\binom{k}{2} - \operatorname{ex}(k, K_{s+1})} + 1 \right\rfloor \\ &= \left\lfloor \frac{r - \binom{k}{2}}{\binom{k}{2} - \operatorname{ex}(k, K_{s+1})} \right\rfloor + 1. \end{aligned}$$

Then

$$z(y) = \left\lfloor \frac{r - \binom{k}{2}}{\binom{k}{2} - y} \right\rfloor + 1 \leq \left\lfloor \frac{r - \binom{k}{2}}{\binom{k}{2} - \operatorname{ex}(k, K_{s+1})} \right\rfloor + 1 = t(s,1)$$

holds if $y \leq \operatorname{ex}(k, K_{s+1})$.

In Appendix A.7 the source code is given to solve LP (37). The optimal value of LP (37) is the same as the optimal value of LP (34) and the best solution for LP (37) is achieved for $s = k - 1$.

This concludes the attempts of generalizing the constraints from LP (17), since other known approaches are equivalent to these presented before. This includes ideas like using Proposition 5.3 and Proposition 5.5 as constraints with both y and s outside the linear program or other variations with parameters outside the LP. It is still possible that a linear program based on the ideas of Proposition 4.3 and Proposition 4.4 could improve the results of LP (34), but this is unlikely.

Comparing the improved results in Table 5 with the upper bounds in Table 1 shows that there is a big gap between the calculated results for LP (34), which is a lower bound, and the upper bounds. As the ideas for constraints based on Proposition 4.3 and Proposition 4.4 do not seem to bring any further improvement, it might be more helpful to consider constraints based on different ideas. One promising idea is presented in the next section.

5.2.3. Forbidden Complete Subgraph Constraint

In this section an idea is presented that could be developed into a constraint. Only the basic idea is presented because the full proposition with the resulting constraint is currently unknown.

The idea is that a multicolored cluster graph generated from a $(K_k, \overline{R})$-free coloring of a graph cannot contain a complete subgraph of certain size if all edges of the subgraph have a given list size. This is formally stated in Lemma 5.7.

Lemma 5.7. *Let $\mathcal{H}$ be the multicolored cluster graph associated with a $(K_k, \overline{R})$-free r-coloring of a graph G, as in the proof of Lemma 4.1. Let $f \geq k$ and $o \leq r$ be positive integers. Then $\mathcal{H}$ is K_f-free if every edge of $\mathcal{H}$ has a list size of o such that*

$$r < \begin{cases} (f-1) \cdot o & \text{if } k = 3 \text{ and } f \text{ is even} \\ f \cdot o & \text{if } k = 3 \text{ and } f \text{ is odd} \\ \binom{f}{2} \cdot o & \text{if } k \geq 4. \end{cases}$$

Proof. Suppose for a contradiction that $\mathcal{H}$ has a K_f subgraph.

First, $k = 3$ is considered. Similar to the proof of Proposition 3.2, the maximum matching of a graph is used. From the proof of Proposition 3.2, it is known that the K_f has $f - 1$ and f disjoint maximum matchings for f even and f odd, respectively. If each matching is assigned a different color list of size o, then every pair of adjacent edges has disjoint color lists. By the choice of o, the set of colors cannot be partitioned into $f - 1$ and f disjoint subsets of size o for f even and f odd, respectively. Therefore, at least two adjacent edges have the same color in their color list, thus allowing a non-rainbow colored K_3, a contradiction.

Consider $k \geq 4$. Fix an edge $e \in E(K_f)$. The color list of this edge needs to be disjoint from the color list of every other edge, as otherwise there is a K_k subgraph of K_f that contains these two edges, thus producing a non-rainbow colored K_k. The existence of such a K_k can be shown by considering that the at most 4 vertices incident with the two edges can be chosen arbitrarily, as $k \geq 4$. This argument holds for every edge of the K_f. Therefore, the color lists of the edges are pairwise disjoint, which means that $\binom{f}{2}$ disjoint subsets of size o are needed. As $r < \binom{f}{2} \cdot o$, at least two color lists contain the same color. This generates a non-rainbow colored K_k, a contradiction. □

In Lemma 5.7 the edges with a single list size are restricted, but the edges with other list sizes are unrestricted. Therefore, using a constraint directly derived from this lemma is not useful for improving the linear program. However, Lemma 5.7 serves as the basic idea for the rest of this section. In Proposition 5.8 a more general version of the lemma is stated, which could be used as a constraint in a linear program.

Proposition 5.8. *Consider a $(K_k, \overline{R})$-free multicolored cluster graph $\mathcal{H}$. Let $f \geq k$ and*

$$o(k, f) \coloneqq \begin{cases} \left\lfloor \frac{r}{f-1} \right\rfloor + 1 & \text{if } k = 3 \text{ and } f \text{ is even} \\ \left\lfloor \frac{r}{f} \right\rfloor + 1 & \text{if } k = 3 \text{ and } f \text{ is odd} \\ \left\lfloor \frac{r}{\binom{f}{2}} \right\rfloor + 1 & \text{if } k \geq 4. \end{cases}$$

Then the multicolored cluster graph with edge set

$$\bigcup_{j=o(k,f)}^{r} e_j(\mathcal{H})$$

is K_f-free.

Proof. For a contradiction suppose that there is a K_f subgraph of $\mathcal{H}$. All the edges of the K_f are in $\bigcup_{j=o(k,f)}^{r} e_j(\mathcal{H})$. If all the edges of the set $\bigcup_{j=o(k,f)}^{r} e_j(\mathcal{H})$ are of one size, then the statement of the proposition is equal to Lemma 5.7.

First, $k = 3$ is considered. If f is even, then the sum of the list sizes for every possible combination of $f-1$ edges is at least $(f-1) \cdot o(k,f) > r$. If f is odd, then the sum of list sizes for every possible combination of f edges is at least $f \cdot o(k,f) > r$. By using the same argument about the maximum matching as in the proof of Lemma 5.7, there is at least one color in at least two lists of adjacent edges. This produces a non-rainbow colored K_3 subgraph of K_f, a contradiction.

Consider $k \geq 4$. As shown in the proof of Lemma 5.7, all color lists of the edges of the K_f must be disjoint to avoid a non-rainbow colored K_k subgraph of K_f. Summing the list sizes of the $\binom{f}{2}$ edges of the K_f gives at least $\binom{f}{2} \cdot o(k,f) > r$. Thus, there is at least one color in at least two lists, which generates a non-rainbow colored K_k subgraph of K_f, a contradiction. □

For $f \geq k$, Proposition 5.8 leads to the inequality

$$\sum_{j=o(k,f)}^{r} e_j(\mathcal{H}) \leq \operatorname{ex}(m, K_f). \tag{38}$$

A linear program could be formulated by using

$$\zeta(\mathcal{H}) = (\operatorname{ex}(m, K_k) - e_{o'(k)}(\mathcal{H}) - \ldots - e_r(\mathcal{H}))/m^2. \tag{39}$$

By Claim 3.8 and (13), the function $o'(k)$ needs to be at least $\lceil r/\ell \rceil$ to guarantee that $\zeta(\mathcal{H}) > \xi$. Furthermore, the coefficients of the variables $e_{o'(k)}(\mathcal{H}), \ldots, e_r(\mathcal{H})$ need to be 1 for all the constraints. Since $o(k,f)$ is maximal for $f = k$, all the coefficients are 1 if $o'(k) \geq o(k,k)$. Thus, the function $o'(k)$ is

$$o'(k) \coloneqq \max\left\{ \left\lceil \frac{r}{\ell} \right\rceil, o(k,k) \right\},$$

as the maximum satisfies both conditions. Since

$$o(k,k) = \left\lfloor \frac{r}{\binom{k}{2}} \right\rfloor + 1 \leq \left\lceil \frac{r}{\ell} \right\rceil,$$

$o'(k)$ is equal to $\lceil r/\ell \rceil$, thus $\lceil r/\ell \rceil$ could be used instead of $o'(k)$.

With $x_i = e_i(\mathcal{H})/(\zeta(\mathcal{H})m^2)$ and (39), the inequality (38) is

$$\begin{aligned}
\sum_{j=o(k,f)}^{r} e_j(\mathcal{H}) &\leq \operatorname{ex}(m, K_f) \\
\sum_{j=o(k,f)}^{o'(k)-1} \zeta(\mathcal{H})m^2 \cdot x_j + \sum_{q=o'(k)}^{r} e_q(\mathcal{H}) &\leq \operatorname{ex}(m, K_f) \\
\zeta(\mathcal{H})m^2 \cdot \sum_{j=o(k,f)}^{o'(k)-1} x_j &\leq \operatorname{ex}(m, K_f) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H}) \\
\left(\operatorname{ex}(m, K_k) - \sum_{i=o'(k)}^{r} e_i(\mathcal{H}) \right) \cdot \sum_{j=o(k,f)}^{o'(k)-1} x_j &\leq \operatorname{ex}(m, K_f) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H}) \\
\sum_{j=o(k,f)}^{o'(k)-1} x_j &\leq \frac{\operatorname{ex}(m, K_f) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H})}{\operatorname{ex}(m, K_k) - \sum_{i=o'(k)}^{r} e_i(\mathcal{H})}.
\end{aligned} \tag{40}$$

Since the numerator of the right hand side of (40) is not $\operatorname{ex}(m, K_k) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H})$ like before, the variables $e_{o'(k)}, \ldots, e_r(r)$ cannot be eliminated. Having $e_r(\mathcal{H})$ as a variable is a problem because the trivial choice of

$$\zeta(\mathcal{H}) = \operatorname{ex}(m, K_k)/m^2$$

leads to $Y_k(r) \geq r$, as the coefficient of $e_r(\mathcal{H})$ is 1 and the right hand side of the resulting constraint is at least 1. Therefore, at least the variable $e_r(\mathcal{H})$ needs to be eliminated to have an r such that $Y_k(r) < r$.

To produce a valid constraint, an upper bound on

$$\frac{\operatorname{ex}(m, K_f) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H})}{\operatorname{ex}(m, K_k) - \sum_{i=o'(k)}^{r} e_i(\mathcal{H})} \tag{41}$$

is needed that does not include the variable $e_r(\mathcal{H})$. The upper bound is required, as the constraint is less restrictive for an upper bound and more restrictive for a lower bound, and a more restrictive constraint results in a smaller optimal value and bigger $r_0(k)$ than actually possible. However, there is no good upper bound known for (41).

If a lower bound on (41) is used, then the resulting value of $r_0(k)$ is an upper bound on $r_0(k)$ that can be achieved with inequality (38). The lower bound

$$\frac{\operatorname{ex}(m, K_f) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H})}{\operatorname{ex}(m, K_k) - \sum_{i=o'(k)}^{r} e_i(\mathcal{H})} \geq \frac{\operatorname{ex}(m, K_f)}{\operatorname{ex}(m, K_k)}$$

leads to the linear program

$$\begin{aligned} \max \quad & x_2 \ln 2 + \cdots + x_{o'(k)-1} \ln\big(o'(k) - 1\big) \\ \text{s.t.} \quad & \sum_{j=o(k,f)}^{o'(k)-1} x_j \leq \frac{\operatorname{ex}(m, K_f)}{\operatorname{ex}(m, K_k)}, \quad f = k, \ldots, r \\ & x_2, \ldots, x_{o'(k)-1} \geq 0. \end{aligned} \tag{42}$$

For n large, the right side of the constraint is

$$\frac{\operatorname{ex}(m, K_f)}{\operatorname{ex}(m, K_k)} \approx \frac{\frac{(f-2)n^2}{(f-1)2}}{\frac{(k-2)n^2}{(k-1)2}} = \frac{(f-2)(k-1)}{(f-1)(k-2)},$$

thus it only depends on k and f. The choice of r as the upper bound on f is to guarantee that at least one constraint restricts the variable x_2. The upper bound on f could be chosen smaller, as the constraint with the smallest f among all the constraints that have a coefficient for all variables of LP (42) has the smallest right hand side, thus it is most restrictive.

The source code to solve LP (42) is stated in Appendix A.8. The optimal value of LP (42) is better than LP (34). However, as the resulting $r_0(k)$ is an upper bound and the actual $r_0(k)$ achieved from inequality (40) could be smaller, it is not really helpful to compare the $r_0(k)$ obtained from LP (42) to the $r_0(k)$ obtained from other linear programs. The value of $r_0(k)$ obtained from LP (42) for $3 \leq k \leq 5$ is stated in Table 7 and the solutions show that $r_0(k)$ for $k \geq 6$ is at least $2 \cdot 10^6$.

Using the knowledge that constraints with smaller f are more restrictive, it is possible to solve LP (42) for bigger r by using only the constraints with the smallest f for each $o(k, f)$, as this reduces the number of constraints in the linear program. Therefore, it is possible to calculate $r_0(6)$, which is also stated in Table 7. The source code to solve this linear program with fewer constraints is given in Appendix A.9.

k	3	4	5	6
$r_0(k)$	12	750	42 372	3 546 116

Table 7: Value of $r_0(k)$ for $3 \leq k \leq 6$ resulting from LP (42). This is an upper bound on $r_0(k)$ that could be obtained from a linear program with inequality (40).

Since the $r_0(k)$ resulting from LP (42) is only an upper bound, LP (42) is not an improvement compared to LP (34). Thus, this linear program is not analyzed further and only the non-zero variables of the optimal solution are given in Table 8 for $3 \leq k \leq 5$.

k	non-zero variables
3	$x_2 \approx 0.1667, x_4 \approx 0.5, x_5 = 1.0$
4	$x_2 \approx 0.0126, x_3 \approx 0.0108, x_4 \approx 0.0093, x_5 \approx 0.0055, x_6 = 0.00625, x_7 \approx 0.0071,$ $x_8 \approx 0.0082, x_9 \approx 0.0096, x_{11} \approx 0.0114, x_{13} \approx 0.0136, x_{16} \approx 0.0167, x_{20} \approx 0.0208,$ $x_{26} \approx 0.0268, x_{35} \approx 0.0357, x_{50} = 0.05, x_{75} = 0.075, x_{125} = 0.125, x_{149} = 1$
5	$x_2 \approx 0.0015, x_3 \approx 0.0012, x_4 \approx 0.0011, x_5 \approx 0.0009, x_6 \approx 0.0009, x_7 \approx 0.0008,$ $x_8 \approx 0.0008, x_9 \approx 0.0007, x_{10} \approx 0.0007, x_{11} \approx 0.0007, x_{12} \approx 0.0006, x_{13} \approx 0.0006,$ $x_{14} \approx 0.0007, x_{15} \approx 0.0005, x_{16} \approx 0.0005, x_{17} \approx 0.0005, x_{18} \approx 0.0006, x_{19} \approx 0.0006,$ $x_{20} \approx 0.0003, x_{21} \approx 0.0007, x_{22} \approx 0.0004, x_{23} \approx 0.0007, x_{24} \approx 0.0004, x_{25} \approx 0.0004,$ $x_{26} \approx 0.0004, x_{27} \approx 0.0004, x_{28} \approx 0.0004, x_{29} \approx 0.0005, x_{30} \approx 0.0005, x_{31} \approx 0.0005,$ $x_{33} \approx 0.0005, x_{34} \approx 0.0005, x_{36} \approx 0.0006, x_{37} \approx 0.0006, x_{39} \approx 0.0006, x_{40} \approx 0.0006,$ $x_{42} \approx 0.0007, x_{44} \approx 0.0007, x_{46} \approx 0.0007, x_{49} \approx 0.0008, x_{51} \approx 0.0008, x_{54} \approx 0.0009,$ $x_{57} \approx 0.0009, x_{60} \approx 0.0009, x_{63} \approx 0.001, x_{67} \approx 0.0011, x_{71} \approx 0.0011, x_{75} \approx 0.0012,$ $x_{80} \approx 0.0013, x_{85} \approx 0.0013, x_{91} \approx 0.0014, x_{97} \approx 0.0015, x_{104} \approx 0.0016,$ $x_{112} \approx 0.0018, x_{120} \approx 0.0019, x_{130} \approx 0.0021, x_{141} \approx 0.0022, x_{153} \approx 0.0024,$ $x_{167} \approx 0.0026, x_{183} \approx 0.0029, x_{201} \approx 0.0032, x_{223} \approx 0.0035, x_{247} \approx 0.0039,$ $x_{276} \approx 0.0044, x_{311} \approx 0.0049, x_{353} \approx 0.0056, x_{403} \approx 0.0063, x_{465} \approx 0.0073,$ $x_{543} \approx 0.0085, x_{642} \approx 0.0101, x_{770} \approx 0.0121, x_{941} \approx 0.0148, x_{1177} \approx 0.0185,$ $x_{1513} \approx 0.0238, x_{2017} \approx 0.0317, x_{2824} \approx 0.0444, x_{4237} \approx 0.0667, x_{4707} = 1$

Table 8: Non-zero variables of the optimal solution of LP (42) for $r = r_0(k)$ and $3 \leq k \leq 5$.

Although LP (42) gives an upper bound on $r_0(k)$, there is still room for improvement. This improvement could be achieved by the idea in this section, since this idea is something like the dual of the calculation for $r_1(k)$. This conjecture is supported by the following LP that does not work and is not based on any proof or formal argument. This constraint is only something that came up by trying out some variations of LP (42) without considering any proof for a corresponding proposition.

The idea of this not working constraint is to use all the constraints from inequality (38) in a single constraint. This requires the function

$$f(k, j) \coloneqq \min \{ f \in \mathbb{N} : f \geq k \wedge j \geq o(k, f) \},$$

which assigns each list size j the minimum f such that $j \geq o(k, f)$. As the right hand side of inequality (38) is different for each value of f, $\operatorname{ex}(m, K_f)$ has to be moved to the left hand side and into the coefficient of the variables to use all constraints in one. This results in the inequality

$$\sum_{j=1}^{r} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,j)})} \cdot e_j(\mathcal{H}) \leq \operatorname{ex}(m, K_k). \tag{43}$$

As $f(k,q) = k$ for $q \geq o'(k)$ and using (39) and (43), the resulting constraint is

$$\begin{aligned}
\sum_{j=2}^{r} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,j)})} \cdot e_j(\mathcal{H}) &\leq \operatorname{ex}(m, K_k) \\
\sum_{j=2}^{o'(k)-1} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,j)})} \cdot \zeta(\mathcal{H}) m^2 \cdot x_j &\leq \operatorname{ex}(m, K_k) - \sum_{q=o'(k)}^{o'(k)-1} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,q)})} \cdot e_q(\mathcal{H}) \\
\zeta(\mathcal{H}) m^2 \cdot \sum_{j=2}^{o'(k)-1} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,j)})} \cdot x_j &\leq \operatorname{ex}(m, K_k) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H}) \\
\sum_{j=2}^{o'(k)-1} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,j)})} \cdot x_j &\leq \frac{\operatorname{ex}(m, K_k) - \sum_{q=o'(k)}^{r} e_q(\mathcal{H})}{\zeta(\mathcal{H}) m^2} \\
\sum_{j=2}^{o'(k)-1} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,j)})} \cdot x_j &\leq 1. \qquad (44)
\end{aligned}$$

The linear program using (44) is

$$\begin{aligned}
\max \quad & x_2 \ln 2 + \cdots + x_{o'(k)-1} \ln\big(o'(k) - 1\big) \qquad (45) \\
\text{s.t.} \quad & \sum_{j=2}^{o'(k)-1} \frac{\operatorname{ex}(m, K_k)}{\operatorname{ex}(m, K_{f(k,j)})} x_j \leq 1 \\
& x_2, \ldots, x_{o'(k)-1} \geq 0.
\end{aligned}$$

In Appendix A.10 the source code is stated to solve this linear program. The optimal solution of LP (45) is better than the result of LP (42) and $r_0(k) = r_1(k) - 1$. However, as LP (45) is invalid, LP (34) remains the best known linear program. The value of $r_0(k)$ for $3 \leq k \leq 4$ is stated in Table 9.

k	3	4
$r_0(k)$	26	759374

Table 9: Value of $r_0(k)$ for $3 \leq k \leq 4$ resulting from LP (45).

Another problem with this LP (45) and the obtained $r_0(k)$ is that for some $r > r_0(k)$ the value of $Y_k(r)$ is smaller than r, which is impossible, as $T_{k-1}(n)$ cannot be $(K_k, \overline{R})$-extremal for $r \geq r_1(k)$. The optimal value of the linear programs that violate this condition are stated in Table 11. This shows that LP (45) is more restrictive than it should be.

The non-zero variables of the optimal solution of LP (45) are given in Table 10.

k	non-zero variables
3	$x_8 = 1.5$
4	$x_{50624} = 1.25$

Table 10: Non-zero variables of the optimal solution of LP (45) for $r = r_0(k)$ and $3 \leq k \leq 4$.

For f in Table 1, which is 4 for $k = 3$ and 6 for $k = 4$, the non-zero variables of the optimal solution of LP (45) have an index of $\lfloor r/(f-1) \rfloor$ and $\left\lfloor r/\binom{f}{2} \right\rfloor$ for $k = 3$ and $k = 4$, respectively. This supports that this approach is somehow similar to the calculation of the

k	$Y_k(r)$ from the linear programs for $r > r_0(k)$ where $r > Y_k(r)$
3	$Y_3(28) = 27, Y_3(29) = 27, Y_3(32) = 31.6228$
4	$Y_4(759376) = 759375, Y_4(759377) = 759375, Y_4(759378) = 759375,$ $Y_4(759379) = 759375, Y_4(759380) = 759375, Y_4(759381) = 759375,$ $Y_4(759382) = 759375, Y_4(759383) = 759375, Y_4(759384) = 759375,$ $Y_4(759385) = 759375, Y_4(759386) = 759375, Y_4(759387) = 759375,$ $Y_4(759388) = 759375, Y_4(759389) = 759375, Y_4(759394) \approx 759393.75,$ $Y_4(759395) \approx 759393.75, Y_4(759396) \approx 759393.75, Y_4(759397) \approx 759393.75,$ $Y_4(759398) \approx 759393.75, Y_4(759399) \approx 759393.75, Y_4(759400) \approx 759393.75,$ $Y_4(759401) \approx 759393.75, Y_4(759402) \approx 759393.75, Y_4(759403) \approx 759393.75,$ $Y_4(759404) \approx 759393.75, Y_4(759412) \approx 759412.5, Y_4(759413) \approx 759412.5,$ $Y_4(759414) \approx 759412.5, Y_4(759415) \approx 759412.5, Y_4(759416) \approx 759412.5,$ $Y_4(759417) \approx 759412.5, Y_4(759418) \approx 759412.5, Y_4(759419) \approx 759412.5,$ $Y_4(759432) \approx 759431.25, Y_4(759433) \approx 759431.25, Y_4(759434) \approx 759431.25$

Table 11: Optimal value of LP (45) for $3 \leq k \leq 4$ where $r > r_0(k)$ and $Y_k(r) < r$.

upper bound $r_1(k)$.

Furthermore, by looking at the optimal values of LP (45) it can be seen that every 3 and 15 consecutive values of r have the same optimal value for $k = 3$ and $k = 4$, respectively. Furthermore, for these r the optimal solution is the same. This indicates that there is an issue with divisibility of r by 3 and 15, which originates from the rounding in the definition of $o(k, f)$. In addition, LP (45) has only one constraint, resulting in exactly one non-zero variable in the optimal solution.

This could be fixed by using more constraints, but at this moment it is unclear how this could be done. One idea might be to generalize the idea of Lemma 5.7 such that all combinations of list sizes with a sum larger than r are considered, rather than using just the list size of a single variable to get the coefficient. However, solving this problem is beyond the scope of this thesis, so this remains an open question that could be explored in the future.

6. Conclusion and Open Questions

In this thesis, it was shown that the Turán graph $T_{k-1}(n)$ is $(K_k, \overline{R})$-extremal for $r \leq r_0(k)$ and cannot be $(K_k, \overline{R})$-extremal for $r \geq r_1(k)$. The value of $r_0(k)$ depends on the optimal value of a linear program that was used as an estimate. Thus, the main focus is to model a linear program such that $r_0(k)$ is as big as possible. Different constraints were tried out and investigated to achieve this. Nevertheless, the gap between $r_0(k)$ and $r_1(k)$ is still very large and it is unknown if the $T_{k-1}(n)$ is $(K_k, \overline{R})$-extremal or not for $r_0(k) < r < r_1(k)$. Furthermore, it would be interesting to known which graph is $(K_k, \overline{R})$-extremal for $r_0(k) < r < r_1(k)$.

This leads to Conjecture 6.1.

Conjecture 6.1. *For $k \geq 3$,*

$$r_0(k) = r_1(k) - 1,$$

where $r_1(k)$ is calculated by the approach in Section 3.2.

Another conjecture is that a Turán graph is $(K_k, \overline{R})$-extremal for any r, which is stated in Conjecture 6.2.

Conjecture 6.2. *For $k \geq 3$, the $(K_k, \overline{R})$-extremal graph for $r \geq r_1(k)$ is the $T_f(n)$ with the maximum lower bound on $|\mathcal{C}_{r,(K_k,\overline{R})}(T_f(n))|$ by the approach in Section 3.2.*

Conjecture 6.1 could be proven by the approach used in this thesis, but this requires an improvement of the considered constraints or new constraints. However, it looks like the ideas of Proposition 4.3 and Proposition 4.4 will not produce significantly better results than LP (34), as all the constraints resulting from these ideas should have been tried. Although, it could be that there is a constraint based on these ideas that proves Conjecture 6.1.

It seems more likely that constraints based on other ideas might prove Conjecture 6.1. The idea in Section 5.2.3 could be such a idea. However, the two presented linear programs using this idea are not correct. Thus, a constraint based on this idea requires more tweaking to make such a linear program work. The most promising starting point to generate a valid constraints seems to be to modify inequality (43) so that it works better with divisibility and uses more than one list size for the K_f.

Nevertheless, it is still possible that there are other ideas for constraints that might work better. It is also possible that a completely different approach, which is not based on a linear program, could prove or disprove Conjecture 6.1.

The focus of this thesis was the extremal coloring problem for the pattern $(K_k, \overline{R})$, but it would be interesting to know if the approach in this thesis can be applied to similar color patterns to obtain a more general result. Furthermore, it may be that the ideas for constraints could also be used for these color patterns. It is possible that color patterns could work where the K_k is colored with at most a certain number of colors. So this could be seen as a generalization of the monochromatic pattern.

References

[17] *RPLEX*, version 0.2, Available at `https://github.com/emallson/rplex`, 2017.

[Alo+04] Noga Alon et al., "The number of edge colorings with no monochromatic cliques", in: *Journal of the London Mathematical Society* 70.2 (2004), pp. 273–288.

[AY06] Noga Alon and Raphael Yuster, "The number of orientations having no fixed tournament", in: *Combinatorica* 26.1 (2006), pp. 1–16.

[Bas+21] Josefran de O. Bastos et al., "Maximum Number of r-Edge-Colorings such that all copies of K_k are rainbow", accepted at LAGOS 2021, 2021.

[BHS17] Fabrício S. Benevides, Carlos Hoppen, and Rudini M. Sampaio, "Edge-colorings of graphs avoiding complete graphs with a prescribed coloring", in: *Discrete Mathematics* 340.9 (2017), pp. 2143–2160.

[Dev20] Rust Project Developers, *The Rust Programming Language*, version 1.45.0, Available at `https://www.rust-lang.org/`, 2020.

[Die05] Reinhard Diestel, *Graph theory*, 3rd, Springer, 2005.

[Erd74] Paul Erdős, *Some new applications of probability methods to combinatorial analysis and graph theory*, University of Calgary, Department of Mathematics, Statistics and Computing Science, 1974.

[ES46] Paul Erdős and Arthur H. Stone, "On the structure of linear graphs", in: *Bull. Amer. Math. Soc* 52.1087-1091 (1946), p. 1.

[Für15] Zoltán Füredi, "A proof of the stability of extremal graphs, Simonovits' stability from Szemerédi's regularity", in: *Journal of Combinatorial Theory, Series B* 115 (2015), pp. 66–71.

[HLN] Carlos Hoppen, Hanno Lefmann, and Denilson Nolibos, "On configurations of generalized Erdős-Rothschild problems", preprint.

[HLO15] Carlos Hoppen, Hanno Lefmann, and Knut Odermann, "A rainbow Erdős-Rothschild problem", in: *Electronic Notes in Discrete Mathematics* 49 (2015), pp. 473–480.

[IBM19] IBM, *IBM ILOG CPLEX User's Manual*, version 12.10.0, Available at `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.10.0/COS_KC_home.html`, 2019.

[KS96] János Komlós and Miklós Simonovits, "Szemerédi's regularity lemma and its applications in graph theory. Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993), 295–352", in: *Bolyai Soc. Math. Stud* 2 (1996).

[Man07] Willem Mantel, "Problem 28 (Solution by H. Gouwentak, W. Mantel, J. Teixeira de Mattes, F. Schuh and W. A. Wythoff)", in: *Wiskundige Opgaven* 10.60-61 (1907), p. 320.

[Rom00] Dan Romik, "Stirling's approximation for n!: The ultimate short proof?", in: *The American Mathematical Monthly* 107.6 (2000), pp. 556–557.

[Sze78] Endre Szemerédi, "Regular partitions of graphs", in: *Problèmes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)* 260 (1978), pp. 399–401.

[Tur41] Paul Turán, "On an extremal problem in graph theory (in Hungarian)", in: *Mat. Fiz. Lapok* 48.436-452 (1941), p. 61.

[Van20] Robert J. Vanderbei, *Linear Programming: Foundations and Extensions*, 5th, Springer International Publishing, 2020.

[Yus96] Raphael Yuster, "The number of edge colorings with no monochromatic triangle", in: *Journal of Graph Theory* 21.4 (1996), pp. 441–452.

Appendices

A. Source Code to solve the Linear Programs

The source code to compute the $r_0(k)$ is written in *Rust*. To solve the linear program the solver *CPLEX* is used. The interface *rplex* is used, to access the *CPLEX* solver from the *Rust* source code.

A.1. Main Function Source Code

The source code of the file `main.rs` to calculate the $r_0(k)$, setup the environment to solve the linear program and solve the linear program. This file is independent of the linear program. The linear program is finally modeled in the file `program.rs`, which depends on the actual linear program.

Listing 1: The `main.rs` source code used to calculate the value of $r_0(k)$.

```
1  mod program;
2
3  use num_integer::binomial;
4  use program::Program;
5  use rplex::*;
6  use std::fs::File;
7  use std::io::Write;
8
9  fn main() {
10     // bounds for k to calculate the r_0(k)
11     let min_vertex_cnt = 3;
12     let max_vertex_cnt = 15;
13     for vertex_cnt in min_vertex_cnt..(max_vertex_cnt + 1) {
14         bin_search(vertex_cnt, false);
15     }
16 }
17
18 fn bin_search(vertex_cnt: usize, save_variables: bool) {
19     println!("\nSolutions␣for␣forbidden␣non-rainbow␣K{}:", vertex_cnt);
20     let mut f = File::create(vertex_cnt.to_string() + ".data").unwrap();
21     // binary search start bounds
22     let mut min_r = 2 * (binomial(vertex_cnt, 2) - 1) + 1;
23     let mut max_r = 10000;
24     for _ in 0..1000 {
25         // CPLEX environment setup
26         let mut env = Env::new().unwrap();
27         env.set_param(EnvParam::Threads(1));
28         let mut prob = Problem::new(&env, "lp").unwrap();
29         let current_r = (min_r + max_r) / 2;
30         let mut prog = Program::new(current_r, vertex_cnt);
31
32         // model linear program
33         prog.add_variables(&mut prob);
34         prog.add_constraints(&mut prob);
```

```
35          prog.add_objective(&mut prob);
36
37          // solve linear program
38          let sol = prob.solve().unwrap();
39
40          // process optimal solution
41          println!(
42              "e^solution␣=␣{}␣for␣r␣=␣{}",
43              std::f64::consts::E.powf(sol.objective),
44              current_r
45          );
46          let output = format!(
47              "{}␣{}\n",
48              current_r,
49              std::f64::consts::E.powf(sol.objective)
50          );
51          match f.write_all(&output.as_bytes()) {
52              Err(error) => eprintln!("Error:␣{}", error),
53              Ok(_) => (),
54          }
55          if save_variables {
56              prog.save_optimal_variables(&sol);
57          }
58
59          // calculate next bounds for binary search or return r_0(k)
60          if (std::f64::consts::E.powf(sol.objective) * 1000000.).round() /
                ↪ 1000000.
61              >= current_r as f64
62          {
63              max_r = current_r;
64          } else {
65              min_r = current_r;
66          }
67          if max_r - min_r <= 1 {
68              println!("For␣K{}␣the␣maximum␣r␣is␣{}", vertex_cnt, min_r);
69              break;
70          }
71      }
72  }
```

A.2. Linear Program (17) Source Code

The file `program.rs` provides the functions to model LP (17). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 2: The `program.rs` source code used to model LP (17).

```
1  use num_bigint::*;
2  use num_integer::binomial;
3  use num_traits::cast::ToPrimitive;
```

```
4  use rplex::*;
5  use std::fs::File;
6  use std::io::Write;
7
8  pub struct Program {
9      r: usize,
10     x: Vec<usize>,
11     k: usize,
12     l: usize,
13 }
14
15 impl Program {
16     pub fn new(r: usize, k: usize) -> Self {
17         return Program {
18             r: r,
19             x: Vec::new(),
20             k: k,
21             l: binomial(k, 2) - 1,
22         };
23     }
24
25     pub fn add_variables(&mut self, prob: &mut Problem) {
26         for i in 2..(self.r - self.l + 1) {
27             let name = i.to_string();
28             self.x.push(
29                 prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
30                     .unwrap(),
31             );
32         }
33     }
34
35     pub fn add_constraints(&mut self, prob: &mut Problem) {
36         // constraint resulting from Proposition 4.3
37         let mut x_weights: Vec<(usize, f64)> = Vec::new();
38         for i in 2..(self.r - self.l + 1) {
39             let weight = (binomial(BigInt::from(self.r), BigInt::from(self.l
                ↪ ))
40                 - binomial(BigInt::from(self.r - i), BigInt::from(self.l)))
41             .to_f64()
42             .unwrap()
43                 / (binomial(BigInt::from(self.r), BigInt::from(self.l)))
44                     .to_f64()
45                     .unwrap();
46             x_weights.push((self.x[i - 2], weight));
47         }
48         prob.add_constraint(con!("subset": 1.0 >= wsum (&x_weights)))
49             .unwrap();
50
51         // constraint resulting from Proposition 4.4
```

```
52          for q in 1..(f64::floor(self.r as f64 / binomial(self.k, 2) as f64)
                ↪ as usize + 1) {
53              let mut x_weights: Vec<(usize, f64)> = Vec::new();
54              for i in usize::max(q, 2)..(self.r - self.l + 1) {
55                  let weight: f64;
56                  if i <= self.r - (self.l * q) {
57                      weight = ((self.k - 2) as f64) / ((self.k - 1) as f64)
58                  } else {
59                      weight = 1.0;
60                  }
61                  x_weights.push((self.x[i - 2], weight));
62              }
63              prob.add_constraint(con!((format!("maxcut: {}", q)): 1.0 >= wsum
                    ↪  (&x_weights)))
64                  .unwrap();
65          }
66      }
67
68      pub fn add_objective(&mut self, prob: &mut Problem) {
69          let mut x_weights: Vec<(usize, f64)> = Vec::new();
70          for i in 2..(self.r - self.l + 1) {
71              let weight = f64::ln(i as f64);
72              x_weights.push((self.x[i - 2], weight));
73          }
74          prob.set_objective(
75              ObjectiveType::Maximize,
76              con!("obj": 1.0 >= wsum (&x_weights)),
77          )
78          .unwrap();
79      }
80
81      pub fn save_optimal_variables(&self, sol: &Solution) {
82          let mut f = File::create(
83              String::from("K")
84                  + self.k.to_string().as_str()
85                  + " r: "
86                  + self.r.to_string().as_str()
87                  + ".vars",
88          )
89          .unwrap();
90          for i in 2..(self.r - self.l + 1) {
91              let output = format!("x{} = {:?}\n", i, sol.variables[i - 2]);
92              match f.write_all(&output.as_bytes()) {
93                  Err(error) => eprintln!("Error: {}", error),
94                  Ok(_) => (),
95              }
96          }
97      }
98  }
```

A.3. Linear Program (29) Source Code

The file `program.rs` provides the functions to model LP (29). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 3: The `program.rs` source code used to model LP (29).

```
use num_bigint::*;
use num_integer::binomial;
use num_traits::cast::ToPrimitive;
use rplex::*;
use std::fs::File;
use std::io::Write;

pub struct Program {
    r: usize,
    x: Vec<usize>,
    k: usize,
    l: usize,
}

impl Program {
    pub fn new(r: usize, k: usize) -> Self {
        return Program {
            r: r,
            x: Vec::new(),
            k: k,
            l: binomial(k, 2) - 1,
        };
    }

    pub fn add_variables(&mut self, prob: &mut Problem) {
        for i in 2..(self.r - self.l + 1) {
            let name = i.to_string();
            self.x.push(
                prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
                    .unwrap(),
            );
        }
    }

    pub fn add_constraints(&mut self, prob: &mut Problem) {
        // constraint resulting from Proposition 5.3
        for y in 1..(self.l + 1) {
            let z = f64::floor(
                (self.r - binomial(self.k, 2)) as f64 / (binomial(self.k, 2)
                    ↪ - y) as f64,
            ) as usize
                + 1;
            let mut x_weights: Vec<(usize, f64)> = Vec::new();
            for i in 2..(self.r - self.l + 1) {
```

```
44                  let weight: f64;
45                  if i <= z {
46                      weight = (binomial(BigInt::from(self.r), BigInt::from(y))
47                          - binomial(BigInt::from(self.r - i), BigInt::from(y))
                                ↪ )
48                      .to_f64()
49                      .unwrap()
50                          / (binomial(BigInt::from(self.r), BigInt::from(y)))
51                              .to_f64()
52                              .unwrap();
53                  } else {
54                      weight = 1.0;
55                  }
56                  x_weights.push((self.x[i - 2], weight));
57              }
58              prob.add_constraint(con!((format!("subset: {}", y)): 1.0 >= wsum
                    ↪  (&x_weights)))
59                  .unwrap();
60          }
61
62          // constraint resulting from Proposition 4.4
63          for q in 1..(f64::floor(self.r as f64 / binomial(self.k, 2) as f64)
                ↪ as usize + 1) {
64              let mut x_weights: Vec<(usize, f64)> = Vec::new();
65              for i in usize::max(q, 2)..(self.r - self.l + 1) {
66                  let weight: f64;
67                  if i <= self.r - (self.l * q) {
68                      weight = ((self.k - 2) as f64) / ((self.k - 1) as f64)
69                  } else {
70                      weight = 1.0;
71                  }
72                  x_weights.push((self.x[i - 2], weight));
73              }
74              prob.add_constraint(con!((format!("maxcut: {}", q)): 1.0 >= wsum
                    ↪  (&x_weights)))
75                  .unwrap();
76          }
77      }
78
79      pub fn add_objective(&mut self, prob: &mut Problem) {
80          let mut x_weights: Vec<(usize, f64)> = Vec::new();
81          for i in 2..(self.r - self.l + 1) {
82              let weight = f64::ln(i as f64);
83              x_weights.push((self.x[i - 2], weight));
84          }
85          prob.set_objective(
86              ObjectiveType::Maximize,
87              con!("obj": 1.0 >= wsum (&x_weights)),
88          )
```

```
89          .unwrap();
90      }
91
92      pub fn save_optimal_variables(&self, sol: &Solution) {
93          let mut f = File::create(
94              String::from("K")
95                  + self.k.to_string().as_str()
96                  + "␣r:␣"
97                  + self.r.to_string().as_str()
98                  + ".vars",
99          )
100         .unwrap();
101         for i in 2..(self.r - self.l + 1) {
102             let output = format!("x{}␣=␣{:?}\n", i, sol.variables[i - 2]);
103             match f.write_all(&output.as_bytes()) {
104                 Err(error) => eprintln!("Error:␣{}", error),
105                 Ok(_) => (),
106             }
107         }
108     }
109 }
```

A.4. Linear Program (32) Source Code

The file `program.rs` provides the functions to model LP (32). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 4: The `program.rs` source code used to model LP (32).

```
1  use num_bigint::*;
2  use num_integer::binomial;
3  use num_traits::cast::ToPrimitive;
4  use rplex::*;
5  use std::fs::File;
6  use std::io::Write;
7
8  pub struct Program {
9      r: usize,
10     x: Vec<usize>,
11     k: usize,
12     l: usize,
13     max_y: usize,
14     max_z: usize,
15 }
16
17 impl Program {
18     pub fn new(r: usize, k: usize) -> Self {
19         let y = binomial(k, 2) - 1;
20         let z = f64::floor((r - binomial(k, 2)) as f64 / (binomial(k, 2) - y
               ↪ ) as f64) as usize + 1;
```

```
21          return Program {
22              r: r,
23              x: Vec::new(),
24              k: k,
25              l: binomial(k, 2) - 1,
26              max_y: y,
27              max_z: z,
28          };
29      }
30
31      pub fn add_variables(&mut self, prob: &mut Problem) {
32          for i in 2..(self.max_z + 1) {
33              let name = i.to_string();
34              self.x.push(
35                  prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
36                      .unwrap(),
37              );
38          }
39      }
40
41      pub fn add_constraints(&mut self, prob: &mut Problem) {
42          // constraint resulting from Proposition 5.3
43          for y in 1..(self.max_y + 1) {
44              let z = f64::floor(
45                  (self.r - binomial(self.k, 2)) as f64 / (binomial(self.k, 2)
                        ↪ - y) as f64,
46              ) as usize
47                  + 1;
48              let mut x_weights: Vec<(usize, f64)> = Vec::new();
49              for i in 2..(self.max_z + 1) {
50                  let weight: f64;
51                  if i <= z {
52                      weight = (binomial(BigInt::from(self.r), BigInt::from(y))
53                          - binomial(BigInt::from(self.r - i), BigInt::from(y))
                                ↪ )
54                      .to_f64()
55                      .unwrap()
56                          / (binomial(BigInt::from(self.r), BigInt::from(y)))
57                              .to_f64()
58                              .unwrap();
59                  } else {
60                      weight = 1.0;
61                  }
62                  x_weights.push((self.x[i - 2], weight));
63              }
64              prob.add_constraint(con!((format!("subset:␣{}", y)): 1.0 >= wsum
                    ↪  (&x_weights)))
65                  .unwrap();
66          }
```

```
67
68          // constraint resulting from Proposition 4.4
69          let lower_bound_q = f64::ceil((self.r - self.max_z) as f64 / self.l
                ↪ as f64) as usize;
70          for q in
71              lower_bound_q..(f64::floor(self.r as f64 / binomial(self.k, 2)
                    ↪ as f64) as usize + 1)
72          {
73              let mut x_weights: Vec<(usize, f64)> = Vec::new();
74              for i in usize::max(q, 2)..(self.max_z + 1) {
75                  let weight: f64;
76                  if i <= self.r - (self.l * q) {
77                      weight = ((self.k - 2) as f64) / ((self.k - 1) as f64)
78                  } else {
79                      weight = 1.0;
80                  }
81                  x_weights.push((self.x[i - 2], weight));
82              }
83              prob.add_constraint(con!((format!("maxcut:␣{}", q)): 1.0 >= wsum
                    ↪  (&x_weights)))
84                  .unwrap();
85          }
86      }
87
88      pub fn add_objective(&mut self, prob: &mut Problem) {
89          let mut x_weights: Vec<(usize, f64)> = Vec::new();
90          for i in 2..(self.max_z + 1) {
91              let weight = f64::ln(i as f64);
92              x_weights.push((self.x[i - 2], weight));
93          }
94          prob.set_objective(
95              ObjectiveType::Maximize,
96              con!("obj": 1.0 >= wsum (&x_weights)),
97          )
98          .unwrap();
99      }
100
101     pub fn save_optimal_variables(&self, sol: &Solution) {
102         let mut f = File::create(
103             String::from("K")
104                 + self.k.to_string().as_str()
105                 + "␣r:␣"
106                 + self.r.to_string().as_str()
107                 + ".vars",
108         )
109         .unwrap();
110         for i in 2..(self.max_z + 1) {
111             let output = format!("x{}␣=␣{:?}\n", i, sol.variables[i - 2]);
112             match f.write_all(&output.as_bytes()) {
```

```
113                 Err(error) => eprintln!("Error:␣{}", error),
114                 Ok(_) => (),
115             }
116         }
117     }
118 }
```

A.5. Linear Program (34) Source Code

The file `program.rs` provides the functions to model LP (34). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 5: The `program.rs` source code used to model LP (34).

```
1  use num_bigint::*;
2  use num_integer::binomial;
3  use num_traits::cast::ToPrimitive;
4  use rplex::*;
5  use std::fs::File;
6  use std::io::Write;
7
8  // exact extremal number
9  fn extremal_number(n: usize, k: usize) -> usize {
10     let all_edges = n / (k - 1);
11     let smaller_parts = binomial(f64::floor(n as f64 / (k - 1) as f64) as
           ↪ usize, 2)
12         * (f64::floor(n as f64 / (k - 1) as f64) as usize * (k - 1) - n + (k
               ↪  - 1));
13     let bigger_parts = binomial(f64::ceil(n as f64 / (k - 1) as f64) as
           ↪ usize, 2)
14         * (n - (f64::floor(n as f64 / (k - 1) as f64) as usize * (k - 1)));
15     return all_edges - smaller_parts - bigger_parts;
16 }
17
18 pub struct Program {
19     r: usize,
20     x: Vec<usize>,
21     k: usize,
22     l: usize,
23 }
24
25 impl Program {
26     pub fn new(r: usize, k: usize) -> Self {
27         return Program {
28             r: r,
29             x: Vec::new(),
30             k: k,
31             l: binomial(k, 2) - 1,
32         };
33     }
```

```
34
35      pub fn add_variables(&mut self, prob: &mut Problem) {
36          for i in 2..(self.r - self.l + 1) {
37              let name = i.to_string();
38              self.x.push(
39                  prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
40                      .unwrap(),
41              );
42          }
43      }
44
45      pub fn add_constraints(&mut self, prob: &mut Problem) {
46          // constraint resulting from Proposition 4.3
47          let mut x_weights: Vec<(usize, f64)> = Vec::new();
48          for i in 2..(self.r - self.l + 1) {
49              let weight = (binomial(BigInt::from(self.r), BigInt::from(self.l
                    ↪ ))
50                  - binomial(BigInt::from(self.r - i), BigInt::from(self.l)))
51              .to_f64()
52              .unwrap()
53                  / (binomial(BigInt::from(self.r), BigInt::from(self.l)))
54                      .to_f64()
55                      .unwrap();
56              x_weights.push((self.x[i - 2], weight));
57          }
58          prob.add_constraint(con!("subset": 1.0 >= wsum (&x_weights)))
59              .unwrap();
60
61          // constraint resulting from Proposition 5.5
62          for s in 1..self.k {
63              for q in 1..(f64::floor(self.r as f64 / binomial(self.k, 2) as
                    ↪ f64) as usize + 1) {
64                  let t = f64::floor(
65                      (self.r - extremal_number(self.k, s + 1) * q) as f64
66                          / (binomial(self.k, 2) - extremal_number(self.k, s +
                                ↪ 1)) as f64,
67                  ) as usize;
68                  let mut x_weights: Vec<(usize, f64)> = Vec::new();
69                  for i in usize::max(q, 2)..(self.r - self.l + 1) {
70                      let weight: f64;
71                      if i <= t {
72                          weight = ((s - 1) as f64) / (s as f64)
73                      } else {
74                          weight = 1.0;
75                      }
76                      x_weights.push((self.x[i - 2], weight));
77                  }
78                  prob.add_constraint(
79                      con!((format!("subset:␣{},␣{}", s, q)): 1.0 >= wsum (&
```

```
                    ↪ x_weights)),
            )
            .unwrap();
        }
    }
}

pub fn add_objective(&mut self, prob: &mut Problem) {
    let mut x_weights: Vec<(usize, f64)> = Vec::new();
    for i in 2..(self.r - self.l + 1) {
        let weight = f64::ln(i as f64);
        x_weights.push((self.x[i - 2], weight));
    }
    prob.set_objective(
        ObjectiveType::Maximize,
        con!("obj": 1.0 >= wsum (&x_weights)),
    )
    .unwrap();
}

pub fn save_optimal_variables(&self, sol: &Solution) {
    let mut f = File::create(
        String::from("K")
            + self.k.to_string().as_str()
            + "␣r:␣"
            + self.r.to_string().as_str()
            + ".vars",
    )
    .unwrap();
    for i in 2..(self.r - self.l + 1) {
        let output = format!("x{}␣=␣{:?}\n", i, sol.variables[i - 2]);
        match f.write_all(&output.as_bytes()) {
            Err(error) => eprintln!("Error:␣{}", error),
            Ok(_) => (),
        }
    }
}
}
```

A.6. Linear Program (35) Source Code

The file `program.rs` provides the functions to model LP (35). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 6: The `program.rs` source code used to model LP (35).

```
use num_bigint::*;
use num_integer::binomial;
use num_traits::cast::ToPrimitive;
use rplex::*;
```

```
5  use std::fs::File;
6  use std::io::Write;
7
8  // exact extremal number
9  fn extremal_number(n: usize, k: usize) -> usize {
10     let all_edges = n / (k - 1);
11     let smaller_parts = binomial(f64::floor(n as f64 / (k - 1) as f64) as
           ↪ usize, 2)
12         * (f64::floor(n as f64 / (k - 1) as f64) as usize * (k - 1) - n + (k
               ↪  - 1));
13     let bigger_parts = binomial(f64::ceil(n as f64 / (k - 1) as f64) as
           ↪ usize, 2)
14         * (n - (f64::floor(n as f64 / (k - 1) as f64) as usize * (k - 1)));
15     return all_edges - smaller_parts - bigger_parts;
16 }
17
18 pub struct Program {
19     r: usize,
20     x: Vec<usize>,
21     k: usize,
22     l: usize,
23     max_y: usize,
24     max_z: usize,
25 }
26
27 impl Program {
28     pub fn new(r: usize, k: usize) -> Self {
29         let y = binomial(k, 2) - 1;
30         let z = f64::floor((r - binomial(k, 2)) as f64 / (binomial(k, 2) - y
               ↪ ) as f64) as usize + 1;
31         return Program {
32             r: r,
33             x: Vec::new(),
34             k: k,
35             l: binomial(k, 2) - 1,
36             max_y: y,
37             max_z: z,
38         };
39     }
40
41     pub fn add_variables(&mut self, prob: &mut Problem) {
42         for i in 2..(self.max_z + 1) {
43             let name = i.to_string();
44             self.x.push(
45                 prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
46                     .unwrap(),
47             );
48         }
49     }
```

```
50
51      pub fn add_constraints(&mut self, prob: &mut Problem) {
52          // constraint resulting from Proposition 5.3
53          for y in 1..(self.max_y + 1) {
54              let z = f64::floor(
55                  (self.r - binomial(self.k, 2)) as f64 / (binomial(self.k, 2)
                        ↪ - y) as f64,
56              ) as usize
57                  + 1;
58              let mut x_weights: Vec<(usize, f64)> = Vec::new();
59              for i in 2..(self.max_z + 1) {
60                  let weight: f64;
61                  if i <= z {
62                      weight = (binomial(BigInt::from(self.r), BigInt::from(y))
63                          - binomial(BigInt::from(self.r - i), BigInt::from(y))
                                ↪ )
64                      .to_f64()
65                      .unwrap()
66                          / (binomial(BigInt::from(self.r), BigInt::from(y)))
67                              .to_f64()
68                              .unwrap();
69                  } else {
70                      weight = 1.0;
71                  }
72                  x_weights.push((self.x[i - 2], weight));
73              }
74              prob.add_constraint(con!((format!("subset:␣{}", y)): 1.0 >= wsum
                    ↪  (&x_weights)))
75                  .unwrap();
76          }
77
78          // constraint resulting from Proposition 5.5
79          for s in 1..self.k {
80              for q in 1..(f64::floor(self.r as f64 / binomial(self.k, 2) as
                        ↪ f64) as usize + 1) {
81                  let t = f64::floor(
82                      (self.r - extremal_number(self.k, s + 1) * q) as f64
83                          / (binomial(self.k, 2) - extremal_number(self.k, s +
                                ↪ 1)) as f64,
84                  ) as usize;
85                  if t > self.max_z {
86                      continue;
87                  }
88                  let mut x_weights: Vec<(usize, f64)> = Vec::new();
89                  for i in usize::max(q, 2)..(self.max_z + 1) {
90                      let weight: f64;
91                      if i <= t {
92                          weight = ((s - 1) as f64) / (s as f64)
93                      } else {
```

```
94                      weight = 1.0;
95                  }
96                  x_weights.push((self.x[i - 2], weight));
97              }
98              prob.add_constraint(
99                  con!((format!("subset:␣{},␣{}", s, q)): 1.0 >= wsum (&
                        ↪ x_weights)),
100             )
101             .unwrap();
102         }
103     }
104 }
105
106 pub fn add_objective(&mut self, prob: &mut Problem) {
107     let mut x_weights: Vec<(usize, f64)> = Vec::new();
108     for i in 2..(self.max_z + 1) {
109         let weight = f64::ln(i as f64);
110         x_weights.push((self.x[i - 2], weight));
111     }
112     prob.set_objective(
113         ObjectiveType::Maximize,
114         con!("obj": 1.0 >= wsum (&x_weights)),
115     )
116     .unwrap();
117 }
118
119 pub fn save_optimal_variables(&self, sol: &Solution) {
120     let mut f = File::create(
121         String::from("K")
122             + self.k.to_string().as_str()
123             + "␣r:␣"
124             + self.r.to_string().as_str()
125             + ".vars",
126     )
127     .unwrap();
128     for i in 2..(self.max_z + 1) {
129         let output = format!("x{}␣=␣{:?}\n", i, sol.variables[i - 2]);
130         match f.write_all(&output.as_bytes()) {
131             Err(error) => eprintln!("Error:␣{}", error),
132             Ok(_) => (),
133         }
134     }
135 }
136 }
```

A.7. Linear Program (37) Source Code

The file `program.rs` provides the functions to model LP (37). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 7: The `program.rs` source code used to model LP (37).

```
use num_bigint::*;
use num_integer::binomial;
use num_traits::cast::ToPrimitive;
use rplex::*;
use std::fs::File;
use std::io::Write;

// exact extremal number
fn extremal_number(n: usize, k: usize) -> usize {
    let all_edges = n / (k - 1);
    let smaller_parts = binomial(f64::floor(n as f64 / (k - 1) as f64) as usize, 2)
        * (f64::floor(n as f64 / (k - 1) as f64) as usize * (k - 1) - n + (k - 1));
    let bigger_parts = binomial(f64::ceil(n as f64 / (k - 1) as f64) as usize, 2)
        * (n - (f64::floor(n as f64 / (k - 1) as f64) as usize * (k - 1)));
    return all_edges - smaller_parts - bigger_parts;
}

pub struct Program {
    r: usize,
    x: Vec<usize>,
    k: usize,
    l: usize,
    max_s: usize,
    max_t: usize,
}

impl Program {
    pub fn new(r: usize, k: usize) -> Self {
        let s = k - 1;
        let t = f64::floor(
            (r - extremal_number(k, s + 1) * 1) as f64
                / (binomial(k, 2) - extremal_number(k, s + 1)) as f64,
        ) as usize;
        return Program {
            r: r,
            x: Vec::new(),
            k: k,
            l: binomial(k, 2) - 1,
            max_s: s,
            max_t: t,
        };
    }

    pub fn add_variables(&mut self, prob: &mut Problem) {
        for i in 2..(self.max_t + 1) {
```

```
46              let name = i.to_string();
47              self.x.push(
48                  prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
49                      .unwrap(),
50              );
51          }
52      }
53
54      pub fn add_constraints(&mut self, prob: &mut Problem) {
55          // constraint resulting from Proposition 5.3
56          for y in 1..(extremal_number(self.k, self.max_s + 1) + 1) {
57              let z = f64::floor(
58                  (self.r - binomial(self.k, 2)) as f64 / (binomial(self.k, 2)
                        ↪ - y) as f64,
59              ) as usize
60                  + 1;
61              let mut x_weights: Vec<(usize, f64)> = Vec::new();
62              for i in 2..(self.max_t + 1) {
63                  let weight: f64;
64                  if i <= z {
65                      weight = (binomial(BigInt::from(self.r), BigInt::from(y))
66                          - binomial(BigInt::from(self.r - i), BigInt::from(y))
                                ↪ )
67                      .to_f64()
68                      .unwrap()
69                          / (binomial(BigInt::from(self.r), BigInt::from(y)))
70                              .to_f64()
71                              .unwrap();
72                  } else {
73                      weight = 1.0;
74                  }
75                  x_weights.push((self.x[i - 2], weight));
76              }
77              prob.add_constraint(con!((format!("subset:␣{}", y)): 1.0 >= wsum
                    ↪  (&x_weights)))
78                  .unwrap();
79          }
80
81          // constraint resulting from Proposition 5.5
82          for s in 1..(self.max_s + 1) {
83              for q in 1..(f64::floor(self.r as f64 / binomial(self.k, 2) as
                    ↪ f64) as usize + 1) {
84                  let t = f64::floor(
85                      (self.r - extremal_number(self.k, s + 1) * q) as f64
86                          / (binomial(self.k, 2) - extremal_number(self.k, s +
                                ↪ 1)) as f64,
87                  ) as usize;
88                  let mut x_weights: Vec<(usize, f64)> = Vec::new();
89                  for i in usize::max(q, 2)..(self.max_t + 1) {
```

```
90                  let weight: f64;
91                  if i <= t {
92                      weight = ((s - 1) as f64) / (s as f64)
93                  } else {
94                      weight = 1.0;
95                  }
96                  x_weights.push((self.x[i - 2], weight));
97              }
98              prob.add_constraint(
99                  con!((format!("subset: {},  {}", s, q)): 1.0 >= wsum (&
                        ↪ x_weights)),
100             )
101             .unwrap();
102         }
103     }
104 }
105
106 pub fn add_objective(&mut self, prob: &mut Problem) {
107     let mut x_weights: Vec<(usize, f64)> = Vec::new();
108     for i in 2..(self.max_t + 1) {
109         let weight = f64::ln(i as f64);
110         x_weights.push((self.x[i - 2], weight));
111     }
112     prob.set_objective(
113         ObjectiveType::Maximize,
114         con!("obj": 1.0 >= wsum (&x_weights)),
115     )
116     .unwrap();
117 }
118
119 pub fn save_optimal_variables(&self, sol: &Solution) {
120     let mut f = File::create(
121         String::from("K")
122             + self.k.to_string().as_str()
123             + "  r:  "
124             + self.r.to_string().as_str()
125             + ".vars",
126     )
127     .unwrap();
128     for i in 2..(self.max_t + 1) {
129         let output = format!("x{}  ={  {:?}\n", i, sol.variables[i - 2]);
130         match f.write_all(&output.as_bytes()) {
131             Err(error) => eprintln!("Error:  {}", error),
132             Ok(_) => (),
133         }
134     }
135 }
136 }
```

A.8. Linear Program (42) Source Code

The file `program.rs` provides the functions to model LP (42). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 8: The `program.rs` source code used to model LP (42).

```
use num_integer::binomial;
use rplex::*;
use std::fs::File;
use std::io::Write;

fn o_function(r: usize, k: usize, f: usize) -> usize {
    let o: usize;
    if k == 3 {
        if f % 2 == 0 {
            o = (r / (f - 1)) + 1;
        } else {
            o = (r / f) + 1;
        }
    } else {
        o = (r / binomial(f, 2)) + 1;
    }
    return o;
}

pub struct Program {
    r: usize,
    x: Vec<usize>,
    k: usize,
    l: usize,
}

impl Program {
    pub fn new(r: usize, k: usize) -> Self {
        return Program {
            r: r,
            x: Vec::new(),
            k: k,
            l: binomial(k, 2) - 1,
        };
    }

    pub fn add_variables(&mut self, prob: &mut Problem) {
        for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
            let name = i.to_string();
            self.x.push(
                prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
                    .unwrap(),
            );
        }
```

```
45      }
46
47      pub fn add_constraints(&mut self, prob: &mut Problem) {
48          // constraint resulting from Proposition 5.8
49          for f in (self.k)..(self.r + 1) {
50              let o = o_function(self.r, self.k, f);
51              let mut x_weights: Vec<(usize, f64)> = Vec::new();
52              for i in o..(f64::ceil(self.r as f64 / self.l as f64) as usize)
                    ↪ {
53                  x_weights.push((self.x[i - 2], 1.0))
54              }
55              let right_side = ((f - 2) * (self.k - 1)) as f64 / ((f - 1) * (
                    ↪ self.k - 2)) as f64;
56              prob.add_constraint(con!(
57                  (format!("first, {}", f)): right_side >= wsum(&x_weights)
58              ))
59              .unwrap();
60              if o <= 2 {
61                  break;
62              }
63          }
64      }
65
66      pub fn add_objective(&mut self, prob: &mut Problem) {
67          let mut x_weights: Vec<(usize, f64)> = Vec::new();
68          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
69              let weight = f64::ln(i as f64);
70              x_weights.push((self.x[i - 2], weight));
71          }
72          prob.set_objective(
73              ObjectiveType::Maximize,
74              con!("obj": 1.0 >= wsum (&x_weights)),
75          )
76          .unwrap();
77      }
78
79      pub fn save_optimal_variables(&self, sol: &Solution) {
80          let mut f = File::create(
81              String::from("K")
82                  + self.k.to_string().as_str()
83                  + " r: "
84                  + self.r.to_string().as_str()
85                  + ".vars",
86          )
87          .unwrap();
88          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
89              let output = format!("x{} = {:?}\n", i, sol.variables[i - 2]);
90              match f.write_all(&output.as_bytes()) {
91                  Err(error) => eprintln!("Error: {}", error),
```

```
                Ok(_) => (),
            }
        }
    }
}
```

A.9. Linear Program (42) Source Code with fewer constraints

The file `program.rs` provides the functions to model LP (42) with fewer constraints to solve linear programs with bigger r. With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 9: The `program.rs` source code used to model LP (42) with fewer constraints.

```
use num_integer::binomial;
use rplex::*;
use std::fs::File;
use std::io::Write;

fn o_function(r: usize, k: usize, f: usize) -> usize {
    let o: usize;
    if k == 3 {
        if f % 2 == 0 {
            o = (r / (f - 1)) + 1;
        } else {
            o = (r / f) + 1;
        }
    } else {
        o = (r / binomial(f, 2)) + 1;
    }
    return o;
}

pub struct Program {
    r: usize,
    x: Vec<usize>,
    k: usize,
    l: usize,
}

impl Program {
    pub fn new(r: usize, k: usize) -> Self {
        return Program {
            r: r,
            x: Vec::new(),
            k: k,
            l: binomial(k, 2) - 1,
        };
    }

```

```
37      pub fn add_variables(&mut self, prob: &mut Problem) {
38          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
39              let name = i.to_string();
40              self.x.push(
41                  prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
42                      .unwrap(),
43              );
44          }
45      }
46
47      pub fn add_constraints(&mut self, prob: &mut Problem) {
48          // constraint resulting from Proposition 5.8 with fewer constraints
49          let mut prev_o = usize::MAX;
50          for f in (self.k)..(self.r + 1) {
51              let o = o_function(self.r, self.k, f);
52              if o == prev_o {
53                  continue;
54              } else {
55                  prev_o = o;
56              }
57              let mut x_weights: Vec<(usize, f64)> = Vec::new();
58              for i in o..(f64::ceil(self.r as f64 / self.l as f64) as usize)
                    ↪ {
59                  x_weights.push((self.x[i - 2], 1.0))
60              }
61              let right_side = ((f - 2) * (self.k - 1)) as f64 / ((f - 1) * (
                    ↪ self.k - 2)) as f64;
62              prob.add_constraint(con!(
63                  (format!("first, {}", f)): right_side >= wsum(&x_weights)
64              ))
65              .unwrap();
66              if o <= 2 {
67                  break;
68              }
69          }
70      }
71
72      pub fn add_objective(&mut self, prob: &mut Problem) {
73          let mut x_weights: Vec<(usize, f64)> = Vec::new();
74          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
75              let weight = f64::ln(i as f64);
76              x_weights.push((self.x[i - 2], weight));
77          }
78          prob.set_objective(
79              ObjectiveType::Maximize,
80              con!("obj": 1.0 >= wsum (&x_weights)),
81          )
82          .unwrap();
83      }
```

```
84
85      pub fn save_optimal_variables(&self, sol: &Solution) {
86          let mut f = File::create(
87              String::from("K")
88                  + self.k.to_string().as_str()
89                  + "␣r:␣"
90                  + self.r.to_string().as_str()
91                  + ".vars",
92          )
93          .unwrap();
94          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
95              let output = format!("x{}␣=␣{:?}\n", i, sol.variables[i - 2]);
96              match f.write_all(&output.as_bytes()) {
97                  Err(error) => eprintln!("Error:␣{}", error),
98                  Ok(_) => (),
99              }
100         }
101     }
102 }
```

A.10. Linear Program (45) Source Code

The file `program.rs` provides the functions to model LP (45). With `main.rs` and `program.rs` the linear program is solved and the $r_0(k)$ is computed for this linear program.

Listing 10: The `program.rs` source code used to model LP (45).

```
1  use num_integer::binomial;
2  use rplex::*;
3  use std::fs::File;
4  use std::io::Write;
5
6  fn o_function(r: usize, k: usize, f: usize) -> usize {
7      let o: usize;
8      if k == 3 {
9          if f % 2 == 0 {
10             o = (r / (f - 1)) + 1;
11         } else {
12             o = (r / f) + 1;
13         }
14     } else {
15         o = (r / binomial(f, 2)) + 1;
16     }
17     return o;
18 }
19
20 pub struct Program {
21     r: usize,
22     x: Vec<usize>,
23     k: usize,
```

```
24      l: usize,
25  }
26
27  impl Program {
28      pub fn new(r: usize, k: usize) -> Self {
29          return Program {
30              r: r,
31              x: Vec::new(),
32              k: k,
33              l: binomial(k, 2) - 1,
34          };
35      }
36
37      pub fn add_variables(&mut self, prob: &mut Problem) {
38          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
39              let name = i.to_string();
40              self.x.push(
41                  prob.add_variable(var!(0.0 <= name <= INFINITY -> 1.0))
42                      .unwrap(),
43              );
44          }
45      }
46
47      pub fn add_constraints(&mut self, prob: &mut Problem) {
48          // constraint resulting from inequality (43)
49          let mut x_weights: Vec<(usize, f64)> = Vec::new();
50          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
51              let mut f = self.k;
52              for f_temp in self.k..self.r {
53                  if i >= o_function(self.r, self.k, f_temp) {
54                      f = f_temp;
55                      break;
56                  }
57              }
58              let weight: f64 = (1. - (1. / (self.k as f64 - 1.))) / (1. - (1.
                    ↪  / (f as f64 - 1.)));
59              x_weights.push((self.x[i - 2], weight));
60          }
61          prob.add_constraint(con!(("complete␣subgraph"): 1.0 >= wsum (&
                ↪ x_weights)))
62              .unwrap();
63      }
64
65      pub fn add_objective(&mut self, prob: &mut Problem) {
66          let mut x_weights: Vec<(usize, f64)> = Vec::new();
67          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
68              let weight = f64::ln(i as f64);
69              x_weights.push((self.x[i - 2], weight));
70          }
```

```
71          prob.set_objective(
72              ObjectiveType::Maximize,
73              con!("obj": 1.0 >= wsum (&x_weights)),
74          )
75          .unwrap();
76      }
77
78      pub fn save_optimal_variables(&self, sol: &Solution) {
79          let mut f = File::create(
80              String::from("K")
81                  + self.k.to_string().as_str()
82                  + "␣r:␣"
83                  + self.r.to_string().as_str()
84                  + ".vars",
85          )
86          .unwrap();
87          for i in 2..(f64::ceil(self.r as f64 / self.l as f64) as usize) {
88              let output = format!("x{}␣=␣{:?}\n", i, sol.variables[i - 2]);
89              match f.write_all(&output.as_bytes()) {
90                  Err(error) => eprintln!("Error:␣{}", error),
91                  Ok(_) => (),
92              }
93          }
94      }
95  }
```

TECHNISCHE UNIVERSITÄT CHEMNITZ

Studentenservice – Zentrales Prüfungsamt

Selbstständigkeitserklärung

Name: Oertel Vorname: Andy geb. am: 31.10.1996 Matr.-Nr.: 406181	**Bitte beachten:** 1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein.

Selbstständigkeitserklärung*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum: 07.05.2021 Unterschrift: ……

* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.