A Comparison of Algorithms for Online Edge-Coloring

Jesper With Mikkelsen
Advisor: Lene Monrad Favrholdt

28. februar 2013
IMADA
Summary

We study the online version of the maximum \( k \)-edge coloring problem. The goal is to color as many edges as possible in a simple graph \( G \) with only \( k \) different colors available. The graph is unknown to the online algorithm which receives the edges one by one. We are particularly interested in comparing the performance of the two previously introduced algorithms First-Fit and Next-Fit. We would also like to know if there are any algorithms which are better than these two.

The main contribution of this report is an analysis of the problem on paths and trees. We show that when the input graph is a path and the algorithm has 2 colors available, then Next-Fit has a competitive ratio of \( \frac{1}{2} \) and First-Fit has a competitive ratio of \( \frac{2}{3} \). The ratio of First-Fit is the best possible ratio for any deterministic algorithm. We present a randomized algorithm which has a competitive ratio of \( \frac{4}{5} \) on paths (against an oblivious adversary) and we prove that this is the best possible ratio for any algorithm. The results are summarized in Table 3.1 on page 26.

When the input graph is restricted to be a tree, we show that any fair algorithm has a competitive ratio which tends to one as \( k \) tends to infinity. The competitive ratio of Next-Fit in this case is \( \frac{2\sqrt{k}-2}{2\sqrt{k}-1} \). This is also the competitive ratio of Next-Fit if we restrict the trees to be \( k \)-colorable. On the other hand, for \( k \)-colorable trees the competitive ratio of First-Fit is between \( \frac{k-2}{k-1} \) and \( \frac{k}{k+1} \). No deterministic algorithm can have a ratio better than \( \frac{k}{k+1} \). We also show that a natural randomized algorithm Rand has a competitive ratio of at most \( \frac{k+3}{k+4} \) on trees. The results are summarized in the table on page 33.

In [FN00], the competitive ratio of Next-Fit and First-Fit on general graphs and on \( k \)-colorable graphs is studied. We show how these results can be extended to cover the competitive ratio of these algorithms on \( \alpha k \)-colorable graphs for some \( \alpha > 0 \). This way of measuring the performance of algorithms is known as the accommodating function. The qualitative results obtained are similar to those in [FN00]. In particular, for \( \frac{1}{2} < \alpha \leq 1 \) we show that the competitive ratio of First-Fit is better than the competitive ratio of Next-Fit for small values of \( k \). But, if one consider the smallest possible ratio for all values of \( k \) they both have a competitive ratio of \( \frac{1}{\alpha k} \). We give an upper bound on the ratio of any deterministic algorithm when \( \alpha \leq 1 \). There is quite a gap between this upper bound and the competitive ratio of First-Fit. However, we do show that on 3-colorable graphs the algorithm First-Fit is optimal (with a competitive ratio of \( \frac{2}{3} \)) among deterministic algorithms when the algorithms have 3 colors available. This particular construction is new. For \( \alpha > 1 \) we completely determine the competitive ratio of Next-Fit. As we will see, the competitive ratio of Next-Fit is always worst possible among fair algorithms. Thus, the value of the accommodating function for Next-Fit
is also a lower bound on the accommodating function for any fair algorithm. The precise results on the accommodating function is given below. For the algorithm \textit{Next-Fit} we show that

\[
\inf_{k \in \mathbb{N}} \{ A_{\text{Next-Fit}}(\alpha,k) \} = \begin{cases} 
1 & \alpha \leq \frac{1}{2} \\
\frac{1}{2\alpha} & \frac{1}{2} \leq \alpha \leq 1 \\
\frac{-\alpha^2+6\alpha-1}{8\alpha} & 1 \leq \alpha \leq \frac{5}{3} \\
\frac{\alpha^2-3\alpha+3}{\alpha} & \frac{5}{3} \leq \alpha \leq \sqrt{3} \\
2\sqrt{3} - 3 & \sqrt{3} \leq \alpha
\end{cases}
\]

Furthermore, if $\frac{1}{2} \leq \alpha \leq 1$ then

\[
A_{\text{First-Fit}}(\alpha,k) = \frac{k}{2\alpha k - 1},
\]

\[
A_{\mathcal{D}}(\alpha,k) \leq \frac{3\alpha - 1}{4\alpha^2 - \alpha},
\]

where \( \mathcal{D} \) is any deterministic algorithm.

The report is structured as follows: In chapter 1, we formally define the problem, performance measures and algorithms under consideration. We also give a brief overview of related work and two useful results on offline edge coloring. The accommodating function is investigated in chapter 2. Some of the general proof techniques and ideas from this chapter will be used in chapter 3 which deals with paths and trees.
1 Introduction

1.1 The max-coloring problem

In the offline version of the max-coloring problem, we are given a fixed number of colors \( k \) and a graph \( G \). The goal is to color as many edges as possible (using only the \( k \) colors) such that no two adjacent edges gets the same color. We will assume throughout that \( G \) is simple and without self-loops, but it may be disconnected. This problem has a natural online version:

**Definition 1.1.** In the *online max-coloring problem*, a fixed number \( k \) of colors is given. The edges of some (unknown) graph is given one by one to the algorithm. Each edge is specified by its endpoints. Immediately upon receiving an edge, the edge must either be colored with one of the \( k \) colors or be rejected. The decision of which color to use is final and can never be changed. Similarly, an rejected edge can never be colored. The goal is to color as many edges as possible.

**Notation and Terminology**

An algorithm for the online max-coloring problem is *fair* if it never rejects an edge unless it is forced to do so.

For convenience, we denote the colors by the numbers 1, 2, \ldots, \( k \). For \( 1 \leq i, j \leq k \), we define \( C_{i,j} \) to be all colors “between” \( i \) and \( j \), that is

\[
C_{i,j} = \begin{cases} 
\{i, i+1, \ldots, j\} & \text{if } i < j \\
\{i, i+1, \ldots, k-1, k, 1, \ldots, j\} & \text{if } j < i
\end{cases}
\]

At any point in the processing of the input sequence \( \sigma \), we denote by \( C_v \) the set of colors used at edges incident to the vertex \( v \). A color \( i \in C_{1,k} \) is said to be available (or missing) at \( v \) if \( i \notin C_v \).

If \( G = (V, E) \) is a graph and \( v \in V \), we denote by \( d(v) \) the degree of \( v \). Furthermore, we let \( \Delta(G) = \max_{v \in V} d(v) \) so that \( \Delta(G) \) is the maximum degree of \( G \). The edge chromatic number \( \chi'(G) \) is defined to be the minimum number of colors needed to make a legal (offline) coloring of all edges in \( G \).

**Performance measures**

Let \( \mathcal{A} \) be an algorithm for the max-coloring problem. If \( \sigma \) is a sequence of edges of some graph, then \( \mathcal{A}_k(\sigma) \) is the number of edges in \( \sigma \) colored by \( \mathcal{A} \) when the algorithm has \( k \) colors available. We denote by \( \text{OPT}_k(\sigma) \) the number of colored edges in an optimal offline solution. Note that \( \text{OPT}_k(\sigma) \) does not depend on the ordering of the edges in the sequence \( \sigma \) but only the resulting graph.
The most used performance measure for online algorithms is the competitive ratio introduced in [KMRS86].

**Definition 1.2.** An algorithm $A$ for the max-coloring problem with $k$ colors available is $c$-competitive if $A_k(\sigma) \geq c \cdot \text{OPT}_k(\sigma)$ for all input sequences $\sigma$ (that is, all possible sequences of the edges of any graph).

The competitive ratio $C_A(k)$ of the algorithm $A$ for the max-coloring problem with $k$ colors available is defined as

$$C_A(k) = \sup \{ c \mid A \text{ is } c\text{-competitive} \} = \sup_{\sigma} \frac{A_k(\sigma)}{\text{OPT}_k(\sigma)}.$$  

Notice that $0 \leq C(A) \leq 1$ and that the better the algorithm, the closer the ratio is to 1. Upper bounds on the competitive ratio come from adversary graphs and lower bounds are performance guarantees.

The above is sometimes referred to as the strict competitive ratio. One then says that an algorithm $A$ is $c$-competitive if there exists a constant $b$ such that $A_k(\sigma) \geq c \cdot \text{OPT}_k(\sigma) - b$. The idea is to prevent some small input sequence which cannot be generalized or extended from influencing the competitive ratio. However, since we do not require our graphs to be connected, the additive constant $b$ usually makes no difference for the max-coloring problem. If one can prove an upper bound by providing some specific adversary graph, multiple copies of the same graph may be given to get past any additive constant. Of course, this only works if the algorithm $A$ makes equivalent colorings on each of the copies, but this will usually be the case.

The competitive ratio sometimes gives overly pessimistic results. Comparing with the best offline solution can make two different algorithms look equally bad even if one of them in practice performs much better. For instance, this is well known to be the case when comparing the competitive ratio of paging algorithms. This has led to the development of many alternative performance measures.

The accommodating function is one such alternative performance measure which generalizes the competitive ratio. For each $\alpha > 0$, one consider the restricted set of input sequences where the optimal offline algorithm does not benefit from having more than $\alpha k$ colors available. It was introduced in [BLN99] for $\alpha \geq 1$ and extended to the case of $\alpha < 1$ in [BFLN02].

**Definition 1.3.** Let $A$ be an online algorithm for the max-coloring problem. For $\alpha > 0$, we define the accommodating function $A_k$ by

$$A_k(\alpha, k) = \sup \{ c \mid A \text{ is } c\text{-competitive on } \alpha k\text{-colorable graphs} \},$$

whenever $\alpha k \in \mathbb{N}$. If this is not the case, then we define $A_k(\alpha, k) = A_k(\lfloor \alpha k \rfloor, k)$.

Note that for $\alpha = 1$, we are considering graphs which are $k$-colorable and that $\lim_{\alpha \to \infty} A_k(\alpha, k) = C_A(k)$. 
Algorithms

In this section, we will define the algorithms that we are going to compare.

Algorithm 1.4. The algorithm Next-Fit uses the color 1 for the very first edge. It always remembers the last color used, \( c_{\text{last}} \). When given a new edge to color, it uses the first available color in the ordered sequence \( \langle c_{\text{last}} + 1, \ldots, k - 1, k, 1, \ldots, c_{\text{last}} \rangle \). If none of the \( k \) colors are available, it rejects the edge.

Next-Fit is a fair algorithm. As the following lemmas show, Next-Fit is the worst possible fair algorithm with respect to the competitive ratio.

Lemma 1.5. If a graph is colored in such a way that each color is used exactly \( n \) or \( n + 1 \) times, then we can make Next-Fit produce this coloring.

Proof. Assume that \( 0 \leq j < k \) of the colors are used \( n + 1 \) times and \( k - j \) of the colors are used \( n \) times. We will make Next-Fit reproduce the coloring using \( C_{1,j} \) for the colors used \( n + 1 \) times and \( C_{j+1,k} \) for the colors used \( n \) times. The adversary gives the edges in \( n + 1 \) rounds. A round consists of giving an edge to be colored 1 followed by an edge to be colored 2 and so on until all \( k \) colors have been used. The last round might be incomplete. In the end Next-Fit will have produced the desired coloring.

We say that two colorings of the same graph are equivalent if one can be obtained from the other by renaming the colors.

Lemma 1.6. Let \( G \) be a colored graph and let \( H \) be a graph consisting of \( k \) copies \( G_1, \ldots, G_k \) of \( G \). It is then possible to make Next-Fit produce a coloring of \( H \) such that each of the \( k \) copies of \( G \) receives a coloring equivalent to the coloring of \( G \).

Proof. We will specify a coloring of \( H \) using each of the \( k \) colors the same number of times. By Lemma 1.5 it is then possible to make Next-Fit produce this coloring. The subgraph \( G_1 \) of \( H \) is colored exactly as \( G \). For \( 1 \leq i < k \), the coloring of \( G_{i+1} \) is obtained from \( G_i \) by shifting the colors once. By this we mean that an edge colored \( j \in C_{1,k} \) in \( G_i \) is colored \( j + 1 \) (or 1 if \( j = k \)) in \( G_{i+1} \).

It follows from Lemma 1.6 that Next-Fit can be made to produce any given coloring of a graph by making multiple copies of the graph. Since we do allow for the input graph to be disconnected, this shows that Next-Fit must have the worst possible competitive ratio of any fair algorithm.

Algorithm 1.7. The algorithm First-Fit uses the color with the lowest possible number whenever it needs to color an edge. If none of the \( k \) colors are available, it rejects the edge.
First-Fit is also a fair algorithm. It is the natural greedy strategy and corresponds to the algorithm called Greedy for the min-coloring problem (see section 1.2).

**Algorithm 1.8.** The algorithm Rand is a randomized algorithm. It selects the color to use for a given edge uniformly at random from all possible colors the edge could receive. It is a fair algorithm which never rejects an edge unless it is forced to do so.

### 1.2 The min-coloring problem

In the min-coloring problem, we wish to use as few different colors as possible. We are not allowed to reject edges but must assign a color (which cannot be changed later) to every single edge when it is revealed.

In [BNMN92] it is shown that the greedy algorithm (which always uses the lowest numbered available color whenever an edge arrives) is optimal with respect to the competitive ratio. For graphs with maximum degree $\Delta$, the greedy algorithm uses at most $2\Delta - 1$ colors since no edge can be adjacent to more than $2\Delta - 1$ other edges. Note that this is true for any algorithm which do not introduce a new color unless forced to do so. On the other hand, even on trees it is possible to make any online algorithm (deterministic or random) use $2\Delta - 1$ colors. Thus, the greedy algorithm is optimal with a competitive ratio of 2.

Note that the greedy algorithm can be seen as a min-coloring version of First-Fit. There is also a natural way to use Next-Fit for the min-coloring problem (which cycles between the colors already used unless it is forced to introduce a new color). Both of these has a competitive ratio of 2, but in [EFKM08] it is shown that First-Fit is better than Next-Fit with respect to another online measure, namely the relative worst-order ratio.

### 1.3 Offline edge coloring

We shall briefly review some classic results on edge coloring of graphs. Recall that $\Delta(G)$ is the maximum degree of $G$ and $\chi'(G)$ is the edge chromatic number. It is clear that at least $\Delta(G)$ colors are always needed to edge color $G$. Vizing’s theorem shows that at most one extra color is needed.

**Theorem 1.9** (Vizing, 1964). Every graph $G$ satisfies

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$$ 

The original proof of the theorem actually gives a polynomial time algorithm for edge coloring a graph using at most $\Delta(G) + 1$ colors. Despite this,
the problem of deciding whether $\chi'(G)$ is equal to $\Delta(G)$ or $\Delta(G) + 1$ is \textbf{NP}-complete. For some classes of graphs, it is known how many colors are needed in an edge coloring. We will often make use of the following result.

\textbf{Theorem 1.10} (König, 1916). \textit{Every bipartite graph $G$ satisfies $\chi'(G) = \Delta(G)$}.

\textbf{Approximating the max edge $k$-coloring problem}

Approximation algorithms for the offline version of maximum edge $k$-coloring has been studied a great deal in recent years for both simple graphs and multigraphs. In what follows, we will briefly describe the results obtained for simple graphs in [FOW02]. It is proven that for every $k \geq 2$ the problem is \textbf{APX}-complete. That is, there exists some constant $\epsilon_k > 0$ such that approximating max edge $k$-coloring within a factor better than $1 - \epsilon_k$ is \textbf{NP}-hard. On the other hand, there do exist polynomial time approximation algorithms with an approximation ratio bounded by a constant factor. One way to do this is to first find a maximum $k$-matching. A $k$-matching is a subgraph for which the degree of each vertex is at most $k$. Note that a maximum $k$-matching must always contain as least as many edges as the number of colored edges in an optimal solution to the max edge $k$-coloring problem. Finding a $k$-matching can be done in polynomial time using an extension of Edmond’s classical blossom algorithm for finding a maximum matching. The $k$-matching can then be colored with at most $k + 1$ colors using the algorithm from Vizing’s Theorem. Removing the least used of the $k + 1$ colors gives a $\frac{k}{k+1}$ approximation. If the graph is bipartite, we know by König’s Theorem that a $k$-matching can be $k$-colored and therefore in this case there is an exact polynomial time algorithm.

For small values of $k$ (in particular for $k$ equal to 2, 3 and 4) there has been several improvements over the $\frac{k}{k+1}$ approximation. For $k = 2$, the best known approximation is $\frac{5}{6}$ and for $k = 3$, the best known approximation is $\frac{13}{15}$. See [Riz09], [Kos09], [KK10] and [CT09] for details.
2 Accommodating function

2.1 The accommodating function for $\alpha \leq 1$

In this section, we will look at the case where the input graph can be colored with $\alpha k$ colors for some $\alpha \leq 1$. In particular, $\text{OPT}$ will be able to color the entire graph. The case when $\alpha = 1$ (i.e. the graph is assumed to be $k$-colorable) has already been investigated in [FN00]. There, the following results were obtained:

**Theorem 2.1.** If $\mathcal{F}$ is fair, then $A_{\mathcal{F}}(1,k) \geq \frac{1}{2}$. If $\mathcal{D}$ is any deterministic algorithm and $k \geq 2$, then $A_{\mathcal{D}}(1,k) \leq \frac{2}{3}$.

**Theorem 2.2.** If $k$ is even, then $A_{\text{Next-Fit}}(1,k) = \frac{1}{2}$. If $k$ is odd, then $A_{\text{Next-Fit}}(1,k) = \frac{1}{2} + \frac{1}{2k^2}$.

**Theorem 2.3.** For all values of $k$ it holds that $A_{\text{First-Fit}}(1,k) = \frac{k}{2k-1}$.

In what follows, we show how the techniques from [FN00] for the case of $k$-colorable graphs can be extended to the case of $\alpha k$-colorable graphs. It turns out that this extension is rather straightforward.

Before we continue, we will introduce some notation which we will use to analyze how an online algorithm performs on a given graph.

**Definition 2.4.** Let $G = (V,E)$ be a graph. Suppose that the online algorithm $\mathcal{A}$ has been given the edges of $G$ (in some sequence). We define $E_c$ to be those edges colored by $\mathcal{A}$ and we define $E_r$ to be those edges rejected by $\mathcal{A}$. For a vertex $x \in V$, we denote by $d_c(x)$ the number of edges in $E_c$ incident to $x$. Likewise, we denote by $d_r(x)$ the number of edges in $E_r$ incident to $x$.

Furthermore, we define $m(x) = \frac{1}{2}d_c(x)$. If $d_r(x) \geq 1$, we also define the function $m_r(x) = \frac{m(x)}{d_r(x)}$.

The idea of Definition 2.4 is the following: Suppose that whenever $\mathcal{A}$ colors an edge $e = (x,y)$, it earns a dollar. For each colored edge, this dollar is then distributed equally among the two endpoints of the edge. The value $m(x)$ tells how many dollars each vertex $x$ receives. This surplus is then distributed equally among the rejected edges incident to $x$. That is, each rejected edge incident to $x$ receives $m_r(x)$ dollars from the vertex $x$. In total, each rejected edge $e = (u,v)$ receives a value of $m_r(u) + m_r(v)$. If we can find a constant $m$ such that each rejected edge receives at least a value of $m$, then we know that $\frac{|E_c|}{|E_r|}$ is at least $m$. This will give us a lower bound of the competitive ratio.

The following lemma will make the above reasoning precise.
THE ACCOMMODATING FUNCTION FOR $\alpha \leq 1$

Lemma 2.5. Let $G = (V, E)$ be a graph and let $A$ be an online algorithm. If $m_r(x) + m_r(y) \geq m$ for all $e = (x, y) \in E_r$ then

$$|E_c| \geq \frac{m}{m+1} |E|.$$ 

Proof. We have that

$$\sum_{(x,y) \in E_r} (m_r(x) + m_r(y)) \leq \sum_{(x,y) \in E_r} (m_r(x) + m_r(y)) + \sum_{d_r(x) = 0} m(x)$$

$$= \sum_{x \in V} m(x)$$

$$= \frac{1}{2} \sum_{x \in V} d_c(x)$$

$$= |E_c|.$$ 

Furthermore, by assumption

$$\sum_{(x,y) \in E_r} (m_r(x) + m_r(y)) \geq m|E_r|.$$ 

It follows that $m|E_r| \leq |E_c|$. Since $|E_r| = |E| - |E_c|$, this is equivalent to

$$\frac{m}{m+1} |E| \leq |E_c|.$$ 

Lemma 2.5 will be used frequently to prove lower bounds on the competitive ratio for $\alpha k$-colorable graphs when $\alpha \leq 1$. Our first application is a lower bound on the accommodating function for fair algorithms when $\frac{1}{2} \leq \alpha \leq 1$.

Theorem 2.6. Let $F$ be a fair algorithm. If $\frac{1}{2} \leq \alpha \leq 1$, then

$$A_F(\alpha, k) \geq \frac{1}{2\alpha}.$$ 

If $\alpha \leq \frac{1}{3}$, then $A_F(\alpha, k) = 1$.

Proof. Consider first the case where $\alpha \leq \frac{1}{3}$. In this case an $\alpha k$-colorable graph has maximum degree at most $\frac{1}{3}k$. It follows that an edge $e$ can be adjacent to at most $2(\frac{1}{3}k - 1) = k - 2$ other edges. Thus, a fair algorithm with $k$ colors available will never reject any edge and the competitive ratio must therefore be 1.

Fix $\frac{1}{2} \leq \alpha \leq 1$ and let $F$ be a fair online algorithm. Consider the coloring produced by $F$ on some $\alpha k$-colorable graph $G = (V, E)$. Since the maximum
degree of $G$ is at most $\alpha k$ and since $F$ never rejects an edge with less than $k$ adjacent colored edges, we get that

\[
d_c(x) + d_r(x) \leq \alpha k, \text{ for } x \in V, \tag{2.6.1}
\]

\[
d_c(x) + d_c(y) \geq k, \text{ for } e = (x,y) \in E_r. \tag{2.6.2}
\]

Let $e = (x,y) \in E_r$. Using (2.6.1) we see that

\[
m_r(x) + m_r(y) = \frac{1}{2} \left( \frac{d_c(x)}{d_r(x)} + \frac{d_c(y)}{d_r(x)} \right) \geq \frac{1}{2} \left( \frac{d_c(x)}{\alpha k - d_c(x)} + \frac{d_c(y)}{\alpha k - d_c(y)} \right).
\]

From (2.6.2) we have $d_c(y) \geq k - d_c(x)$ which in turn gives the following estimate

\[
\alpha k - d_c(y) \leq \alpha k - (k - d_c(x)) = d_c(x) - (1 - \alpha)k.
\]

Inserting this into our lower bound of $m_r(x) + m_r(y)$ we get

\[
m_r(x) + m_r(y) \geq \frac{1}{2} \left( \frac{d_c(x)}{\alpha k - d_c(x)} + \frac{d_c(y)}{\alpha k - d_c(y)} \right) \geq \frac{1}{2} \left( \frac{d_c(x)}{\alpha k - d_c(x)} + \frac{k - d_c(x)}{d_c(x) - (1 - \alpha)k} \right).
\]

This expression depends only on $d_c(x)$ since $\alpha$ and $k$ are fixed. The number of colored edges $d_c(x)$ must be between $(1 - \alpha)k$ and $\alpha k$ since the graph needs to be $\alpha k$-colorable. If we allow $d_c(x)$ to be any real number in this interval, we will get a lower bound which also holds when $d_c(x)$ is required to be an integer. Using calculus, we get that the expression obtains its lowest possible value at $d_c(x) = \frac{k}{2}$. Evaluating at this point gives

\[
m_r(x) + m_r(y) \geq \frac{1}{2} \cdot \frac{1}{2\alpha - 1}.
\]

Notice that this lower bound holds for all rejected edges and does not depend on the particular way the adversary reveals the graph $G$. It only depends on the fact that $F$ is fair and that $G$ can be $\alpha k$-colored. By Lemma 2.5, we get the claimed lower bound for the accommodating function of

\[
\frac{1}{2\alpha - 1} + 1 = \frac{1}{2\alpha}.
\]

At first sight, one may think that the lower bound of $\frac{1}{2\alpha}$ is too pessimistic. We only look at one rejected edge at a time and see how bad it can be for the
online algorithm. However, as the following result shows, the fair algorithm Next-Fit meets this lower bound. The adversary can force Next-Fit into the case where at each vertex, it has $\frac{k}{2}$ colored edges and $\alpha k - \frac{k}{2}$ rejected edges (for $k$ even).

**Theorem 2.7.** If $k$ is even and $\frac{1}{2} \leq \alpha \leq 1$, then

$$A_{NF}(\alpha, k) = \frac{1}{2\alpha}.$$  

**Proof.** Let $G_1 = (L_1 \cup R_1, E_1)$ and $G_2 = (L_2 \cup R_2, E_2)$ be copies of the complete bipartite graph $K_{\frac{k}{2}, \frac{k}{2}}$. We know that this bipartite graph can be $\frac{k}{2}$-colored. By Lemma 1.5, we can make Next-Fit color $E_1$ with the colors $C_{1, \frac{k}{2}}$ and $E_2$ with the colors $C_{\frac{k}{2}+1, k}$. To see why, notice that this coloring of $E_1 \cup E_2$ uses each of the $k$ colors the same number of times.

The adversary now connects each vertex in $R_1$ to $\alpha k - \frac{k}{2}$ vertices in $L_2$ and each vertex in $R_2$ to $\alpha k - \frac{k}{2}$ vertices in $L_1$. None of these added edges can be colored by Next-Fit. However, the graph is a bipartite graph with maximum degree $\alpha k$ and therefore it is $\alpha k$-colorable. This gives the desired ratio of

$$\frac{2\left(\frac{k}{2}\right)^2}{2\left(\frac{k}{2}\right)^2 + 2(\alpha k - \frac{k}{2})\frac{k}{2}} = \frac{1}{2\alpha}.$$  

$\square$

We will now show that First-Fit also has a competitive ratio of $\frac{1}{2\alpha}$ on $\alpha k$-colorable graphs. However, for small values of $k$ First-Fit actually has a better ratio. This is shown by the following lower bound.

**Lemma 2.8.** For $\frac{1}{2} < \alpha \leq 1$ we have that

$$A_{FF}(\alpha, k) \geq \frac{k}{2\alpha k - 1}.$$  

**Proof.** Let $G = (V, E)$ be an $\alpha k$-colorable graph. For $f \in C_{1, k}$, we let $E_f$ be the edges colored $f$ by First-Fit. We will show by induction on $f$ that for all $f \in C_{1, k}$,

$$\sum_{i=1}^{f} |E_i| \geq \frac{f}{2\alpha k - 1} |E|.$$  \hspace{1cm} (2.8.1)

For $f = 1$, we need to show that $|E_1| \geq \frac{|E|}{2\alpha k - 1}$. This is done using two simple observations:

1. An edge in $E \setminus E_1$ is adjacent to at least one edge from $E_1$.
2. An edge in $E_1$ is adjacent to at most $2(\alpha k - 1)$ edges from $E \setminus E_1$. 


Combining these observations, we get that \(|E \setminus E_1| \leq 2(\alpha k - 1)|E_1|\). Since \(|E \setminus E_1| = |E| - |E_1|\), this implies that

\[|E| \leq 2(\alpha k - 1)|E_1| + |E_1| = (2\alpha k - 1)|E_1|.
\]

Now, let \(f \in C_{2,k}\) and assume that the statement (2.8.1) is true for the color \(f - 1\). As in the base case, we use the following observations:

1. An edge in \(E \setminus \bigcup_{i=1}^{f} E_i\) is adjacent to at least one edge from \(E_f\).
2. An edge in \(E_f\) is adjacent to at least \(f - 1\) edges from \(\bigcup_{i=1}^{f-1} E_i\). Thus, an edge in \(E_f\) is adjacent to at most \(2(\alpha k - 1) - (f - 1) = 2\alpha k - f - 1\) edges from \(E \setminus \bigcup_{i=1}^{f} E_i\).

Combining these observations, we get that \(|E \setminus \bigcup_{i=1}^{f} E_i| \leq 2(\alpha k - f - 1)|E_f|\). This implies that \(|E \setminus \bigcup_{i=1}^{f-1} E_i| \leq (2\alpha k - f - 1)|E_f| + |E_f|\) which is equivalent to

\[|E_f| \geq \frac{|E \setminus \bigcup_{i=1}^{f-1} E_i|}{2\alpha k - f}.
\]

We know that \(2\alpha k - f > 0\) since we assumed \(\frac{1}{2} \leq \alpha \). The induction hypothesis can now be used to make the following calculation which proves the theorem,

\[
\sum_{i=1}^{f} |E_i| \geq \sum_{i=1}^{f-1} |E_i| + \frac{|E| - \bigcup_{i=1}^{f-1} E_i|}{2\alpha k - f}
= \frac{|E| + (2\alpha k - f - 1) \sum_{i=1}^{f-1} |E_i|}{2\alpha k - f}
\geq \frac{|E| + (2\alpha k - f - 1) \sum_{i=1}^{f-1} |E|}{2\alpha k - f}
= \frac{f}{2\alpha k - 1}|E|.
\]

Note that the ratio \(\frac{k}{2\alpha k - 1}\) tends to \(\frac{1}{2\alpha}\) when \(k\) tends to infinity. We will now show that the above lower bound is actually tight.

**Lemma 2.9.** If \(\frac{1}{2} \leq \alpha \leq 1\), then

\[\mathcal{A}_{FF}(\alpha, k) \leq \frac{k}{2\alpha k - 1}.
\]

**Proof.** We will use the analysis in the proof of Theorem 2.8. It shows that if no edge is adjacent to more than one edge of each color and if each vertex has degree \(\alpha k\), then First-Fit colors \(\frac{k}{2\alpha k - 1}\) of the edges. For simplicity, we will assume that \(k\) is even. It is not difficult to modify the proof to also work for \(k\) odd.
The construction used in this proof is somewhat complicated. We will first define a family of bipartite graphs which will be used as building blocks. Then, we will show how First-Fit can be made to color these building blocks in a particular way. Finally, we will show how a large number of copies of the building blocks can be connected. This will be done so that First-Fit must reject all edges connecting the building blocks and will give a graph for which the analysis in Theorem 2.8 is tight.

The building blocks: The adversary graph is built from \((\alpha - \frac{1}{2})k\) bipartite, biregular graphs, \(G_1, G_2, \ldots, G_{(\alpha - \frac{1}{2})k}\). Denote by \(X_i, Y_i\) the vertex partition of \(G_i\). We will think of each vertex as corresponding to a certain subset of the \(k\) colors. There is a vertex in \(X_i\) for each subset of \(C_{1,k}\) of size \(\alpha k + 1 - i\) and there is a vertex in \(Y_i\) for each subset of \(C_{1,k}\) of size \((1 - \alpha)k + i\). For a vertex \(v\), we denote by \(C(v)\) the set of colors that \(v\) corresponds to. There is an edge between \(x \in X_i\) and \(y \in Y_i\) whenever \(C(x) \cup C(y) = C_{1,k}\). Notice that for each edge \(e = (x, y)\) in \(G_i\) we have that \(|C(x)| + |C(y)| = \alpha k + 1 - i + (1 - \alpha)k + i = k + 1\) and therefore \(|C(x) \cap C(y)| = 1\).

Coloring of the building blocks: We claim that First-Fit can be forced to color \(G_i\) such that the edge \(e\) receives the color \(C(x) \cap C(y)\). The first thing we need to show is that this is actually a valid coloring. Fix some \(x \in X_i\). For each \(c \in C(x)\), there is exactly one \(y \in Y_i\) such that \(C(x) \cap C(y) = \{c\}\). Conversely, to each \(c \in C(y)\) for some vertex \(y \in Y_i\), there is exactly one \(x \in X_i\) such that \(C(x) \cap C(y) = \{c\}\). We conclude that no two adjacent edges receive the same color. Furthermore, the degree of a vertex \(v\) in \(G_i\) is \(|C(v)|\). We can make First-Fit produce this coloring since each edge \(e = (x, y)\) is adjacent to (exactly) one edge of each color in \(C_{1,k} \setminus (C(x) \cap C(y))\). Thus, the adversary can give the edges that needs to be colored 1 first, then the edges that needs to be colored 2 and so on.

Putting it all together: The adversary uses \((2\alpha - 1)k\) bipartite graphs \(G^L_1, G^L_2, \ldots, G^L_{(\alpha - \frac{1}{2})k}\) and \(G^R_1, G^R_2, \ldots, G^R_{(\alpha - \frac{1}{2})k}\). For \(i = 1, 2, \ldots, (\alpha - \frac{1}{2})k\), the graph \(G^L_i\) is isomorphic to \(G^R_i\). The graph \(G^L_i\) consists of \(n_i\) copies of \(G_i\). The number \(n_i\) is defined by the recursion

\[
n_1 = 1, \quad n_{i+1} = \frac{(2\alpha - 1)k - i}{i} n_i, \quad \text{for } i = 1, 2, \ldots, (\alpha - \frac{1}{2})k - 1.
\]

Fix \(i = 1, 2, \ldots, (\alpha - \frac{1}{2})k - 1\). We will show how the adversary can give edges between \(G^L_i\) and \(G^L_{i+1}\), none of which can be colored by First-Fit. First, note that since \(k - ((1 - \alpha)k + i) = \alpha k - i\), the number of vertices in \(Y_i\) is the same as the number of vertices in \(X_{i+1}\). It also follows that for each vertex \(y \in Y_i\), there is exactly one vertex \(x \in X_{i+1}\) such that \(C(y) \cup C(x) = C_{1,k}\). If two such vertices are connected by an edge, this edge must be rejected.

Now, in order to have every vertex \(y \in Y_i\) get degree \(\alpha k\), we need to connect it to \((2\alpha - 1)k - i\) vertices in \(X_{i+1}\). Similarly, for every vertex \(x \in X_{i+1}\) to
THE ACCOMMODATING FUNCTION FOR $\alpha \leq 1$

get degree $\alpha k$ we need to connect it to $i$ vertices in $Y_i$. This is possible since by the recursive definition of $n_i$, the number of copies of $G_i$ in $G^L_i$ is exactly $\frac{(2\alpha-1)^{k-i}}{i}$ times the number of copies of $G_{i+1}$ in $G^L_{i+1}$.

Finally, the vertices in $Y^L_{(\alpha - \frac{1}{2})k}$ is connected to the vertices in $Y^R_{(\alpha - \frac{1}{2})k}$. This is done by adding edges between pair of vertices $y^L \in Y^L_{(\alpha - \frac{1}{2})k}$ and $y^R \in Y^R_{(\alpha - \frac{1}{2})k}$ where $C(y^L) \cup C(y^R) = C_1k$ in a way so that the degree of all vertices in $Y^L_{(\alpha - \frac{1}{2})k} \cup Y^R_{(\alpha - \frac{1}{2})k}$ becomes $\alpha k$.

**Theorem 2.10.** For $\frac{1}{2} < \alpha \leq 1$ we have that

$$A_{FF}(\alpha, k) = \frac{k}{2\alpha k - 1}.$$

**Proof.** This follows from Lemma 2.8 and 2.9.

We will now give an upper bound on the accommodating function for any deterministic algorithm.

**Theorem 2.11.** Let $\frac{1}{2} \leq \alpha \leq 1$. Then any deterministic algorithm $A$ satisfies

$$A_k(\alpha, k) \leq \frac{3\alpha - 1}{4\alpha^2 - \alpha}.$$

**Proof.** We assume that $k$ is even. The adversary constructs a $\frac{k}{2}$-regular bipartite graph $G = (L, R)$ with $|L| = |R| = N$ for some large $N$. If the algorithm is fair it will color all these edges, but an unfair algorithm might choose to reject some of them. We denote by $C_1, \ldots, C_p$ the $p = \sum_{i=0}^{\frac{k}{2}} \binom{k}{i}$ different possibilities for $C_x$ where $x$ is a vertex in $G$. For $i = 1, \ldots, p$, let $S^L_i$ be those vertices in $L$ where the incident edges are colored using $C_i$, that is

$$S^L_i = \{x \in L \mid C_x = C_i\}.$$ $S^R$ is defined similarly. Now, partition $S^L_i$ into $\lfloor \frac{|S^L_i|}{\alpha k} \rfloor$ subsets of size $\alpha k$ and possibly a single subset of size strictly smaller than $\alpha k$. We make the same partitioning of $S^R_i$. Let $S$ be the family of all these subsets which have size exactly $\alpha k$. Note that in each $S^L_i$, there is at most $\alpha k - 1$ vertices which are not contained in a subset of size $\alpha k$. The same thing is true for vertices from $S^R_i$. Thus, there is at most $2(N - (\alpha k - 1)p)$ vertices not contained in a subset of size $\alpha k$. By making $N$ sufficiently large, we can ignore these vertices and the edges incident to them.

For each $S \in S$, the adversary makes $\alpha k - \frac{k}{2}$ new vertices. The adversary then gives the edges of a complete bipartite graph between these new vertices and $S$. Thus, the adversary gives $(\alpha k - \frac{k}{2})\alpha k$ edges. Note that both the new
vertices and the vertices in $S$ gets degree $\alpha k$. Also, since $\mathcal{F}$ uses the same $d$ colors at each of the vertices in $S$, it can color at most $(\alpha k - \frac{k}{2})(k - d)$ of the added edges. We say at most since $k - d$ might be bigger than $\alpha k$ (even through this is clearly a bad thing for the algorithm). On the other hand, since $S \subseteq L$ or $S \subseteq R$, we see that the graph remains bipartite. It follows that the graph is $\alpha k$ colorable and therefore $\text{OPT}$ can color all edges in the graph.

For each edge $e = (x, y)$ where $x \in S$ and $y \in S'$ for some $S, S' \in S$, we count this edge as half an edge for $S$ and half an edge for $S'$. If we sum over all subsets $S \in \mathcal{S}$, we will count each edge exactly once (since we are working under the assumption that all edges are contained in $\mathcal{S}$). We get a count of $\frac{1}{2}(\alpha k \frac{k}{2}) = \frac{1}{4}\alpha k^2$ from the edges between $L$ and $R$ for each $S$. The algorithm $\mathcal{A}$ gets $\frac{1}{2}(\alpha kd)$ of this value. Thus, for each $S$ the ratio of colored edges to the total number of edges in $S$ is at most

$$\frac{(\alpha k - \frac{k}{2})(k - d) + \frac{1}{2}(\alpha kd)}{(\alpha k - \frac{k}{2})\alpha k + \frac{1}{4}\alpha k^2}.$$  

The derivative of this expression with respect to $d$ is

$$\frac{2(1 - \alpha)}{k\alpha(4\alpha - 1)}.$$  

It follows that the expression is non-decreasing in the variable $d$. In fact, if $\alpha < 1$ it is strictly increasing in $d$. Thus, we get an upper bound for all deterministic algorithms by letting $d = \frac{k}{2}$. In other words, we may assume that the algorithm is fair since it gets a worse competitive ratio by being unfair. This gives us the desired upper bound:

$$\frac{(\alpha k - \frac{k}{2})(\frac{k}{2}) + \frac{1}{4}\alpha k^2}{(\alpha k - \frac{k}{2})\alpha k + \frac{1}{4}\alpha k^2} = \frac{3\alpha - 1}{4\alpha^2 - \alpha}.$$  

There is quite a gap between the competitive ratio of First-Fit on $\alpha k$ colorable graphs for $\alpha \leq 1$ (given in Theorem 2.10) and the upper bound obtained in Theorem 2.11. I therefore tried to look into some specific cases where $k$ is small and $\alpha = 1$. For $k = 2$ and $k = 3$, it turns out that First-Fit is an optimal deterministic algorithm. For $k = 2$, the competitive ratio of First-Fit is $\frac{2}{3}$ and we show in Theorem 3.2 that no deterministic algorithm can do better than this. The adversary graph is a path. For $k = 3$, the following theorem shows that First-Fit is optimal.

**Theorem 2.12.** For any deterministic algorithm $\mathcal{A}$, we have that $\mathcal{A}_\mathcal{A}(1, 3) \leq \frac{2}{5}$. In particular, First-Fit is an optimal deterministic algorithm on 3-colorable graphs.
Proof. From Theorem 2.10 we know that $A_{FF}(1, 3) = \frac{2}{5}$. We will show that this is the best possible competitive ratio for any deterministic algorithm $A$ on 3-colorable graphs.

The adversary constructs a graph depending on which edges (if any) $A$ chooses to reject. We will need to introduce some terminology to describe how the adversary achieves this. The adversary gives a small number of edges (one to four) to construct a widget in each step. We will say that $A$ cannot afford to reject an edge in a widget if this would locally give us a ratio no better than $\frac{3}{5}$. That is, if we would get a connected subgraph $G' = (V', E')$ for which $\frac{|E'|}{|E'|} \leq \frac{3}{5}$. If this happens, the adversary can stop giving more edges to the widget and put it aside. It then tries to make the same widget again. At some point, $A$ must either allow the adversary to construct the desired widget or we will have shown that $A_{A}(1, 3) \leq \frac{3}{5}$.

The graph itself contains a number of copies of three components (denoted component 1, 2 and 3) each consisting of three widgets. The graph also contains the widgets put aside. Not all of the components needs to be present in the graph. A partial component is a subgraph of one of the three (complete) components. The three components are described in the Figure 2.1, 2.2 and 2.3. A dotted line indicates an edge rejected by $A$ being unfair. Below each component, we describe how one can force $A$ to reject a certain number of edges once a component has been constructed. Notice that in each case, we locally get a ratio no better than $\frac{3}{5}$ (since all of the components together with the added edges can be 3-colored). Also, both component 1 and component 3 exists in three different variations depending on the coloring used by $A$. 

![Figure 2.1: Component 1. Four edges can be added none of which can be colored. This gives 8 colored and 6 rejected edges. This gives a ratio of $\frac{8}{14} < \frac{3}{5}$.](image_url)
We now describe how the adversary will construct these components depending on how $A$ colors the edges given. See also Figure 2.4. For the basis step, the widget given is a fan with two or three edges. Notice that $A$ cannot afford to reject neither the first nor the second edge given in the fan. We consider two main cases:

Case (1): If we have two triangles in a partial component 1 where the edges adjacent to the rejected edge is colored with the same two colors as the new fan, then the adversary will complete this partial component.

Case (2): If the widget was not used to complete a partial component 1 (or put aside), the adversary will give all three edges in the fan. There are now two subcases, depending on whether the third edge in the fan is rejected or not.

Case (2a): If the third edge is rejected, the adversary makes a new edge between the two colored edges. Since $A$ cannot afford to reject this edge, we get a triangle using all three colors with a rejected edge at one of the vertices. This is then added to the corresponding partial component 1.

Case (2b): If all three edges are colored, this fan is added to a partial component 2. In this case, the adversary will from now on only give triangles until this partial component 2 has been completed. The algorithm $A$ cannot afford to reject more than one edge from each triangle. If it does not reject any edge, the triangle is added to the partial component 2. If a single edge
begin the construction of a widget $W$ by giving two edges $e_1$ and $e_2$ connected at a single vertex $v$;

if $W$ can be used to complete a partial component 1 then
  put $W$ in the corresponding partial component 1;
  restart;
else
  give a third edge $e_3$ incident to $v$;
  if $e_3$ is rejected then
    give an edge connecting $e_1$ and $e_2$ and add $W$ to the corresponding partial component 1;
    restart;
  else
    add $W$ to a partial component 2;
    while the partial component 2 is not completed do
      give a triangle $T$;
      if all edges of $T$ are colored then
        add $T$ to the partial component 2;
      else
        add $T$ to the corresponding partial component 3;
      end
    end
  end
end

Figure 2.4: An algorithmic description of one step of the adversary strategy used in the proof of Theorem 2.11. For clarity, we have not included the checks of whether or not the algorithm has rejected enough edges to locally get a ratio no better than $\frac{3}{5}$.

is rejected, the triangle is added to the corresponding partial component 3. When the partial component 2 has been completed, the adversary will again give fans instead of triangles and follow the procedure described above.

We will now argue why the construction works. Assume that the algorithm never rejects an edge when it cannot locally afford to do so. Then at each step, the adversary will add a number of edges to some component. We claim that there is only a constant number of edges that can be part of a partial component:

There can be at most three partial components of type 1, one for each of the three possible combinations of colors used at the edges adjacent to the rejected edge in the triangle. Assume that we have a partial component 1 with two triangles where the edges adjacent to the rejected edge is colored with the colors $a, b$. Then there will never be created another partial components of
type 1 with these two colors, until the one we already have created has been completed. This is because the creation of a second partial component of the same type would require the adversary to first make a fan with two edges colored with the colors \( a, b \). But this fan would then immediately be used to complete the partial component 1.

It is clear that there cannot be more than one partial component of type 2 at any time in the construction, since the adversary does not go to the next step before it has completed this partial component. Also, there cannot be more than three partial components of type 3, one for each of the three combinations of colors used in a triangle with two colored edges.

Thus, the number of complete components will become arbitrarily large as the adversary graphs grows and the number of edges not contained in a complete component will be bounded by a constant. Since each complete component locally has a ratio no better than \( \frac{3}{5} \) this gives the desired result.

Notice that if the algorithm \( A \) is fair, we will get a complete component 2 at the very beginning (and no other edges) which the adversary then uses to show that the competitive ratio of \( A \) is at most \( \frac{2}{5} \).

### 2.2 The accommodating function for \( \alpha > 1 \)

We will now look into the accommodating function when \( \alpha > 1 \). In [FN00], the following results are proved:

**Theorem 2.13.** If \( F \) is fair, then for all values of \( \alpha \)

\[
A_F(\alpha, k) \geq \min_{d \in C_{1,k}} \left\{ \frac{k^2 + d^2 - kd}{2k^2 - kd} \right\} \geq 2\sqrt{3} - 3
\]

**Theorem 2.14.** If \( F \) is fair and \( \alpha \geq 2 \), then \( A_F(\alpha, k) \leq \frac{1}{2} \).

**Theorem 2.15.** If \( A \) is any algorithm (possibly randomized and unfair) and \( \alpha \geq 4 \), then \( A_A(\alpha, k) \leq \frac{4}{7} \).

**Theorem 2.16.** If \( \alpha \geq \sqrt{3} \), then

\[
A_{\text{Next-Fit}}(\alpha, k) \leq \min_{d \in C_{1,k}} \left\{ \frac{k^2 + d^2 - kd}{2k^2 - kd} \right\}.
\]

This minimum can become arbitrarily close to \( 2\sqrt{3} - 3 \) if the value of \( k \) is not bounded.

We do not offer any improvements on the upper bounds in Theorem 2.14 and Theorem 2.15. We do however extend Theorem 2.13 and Theorem 2.16 by determining the accommodating function for \( \text{Next-Fit} \) for all values of \( \alpha \). We will first give the upper bounds. It turns out that there are two different cases to consider: One for \( \alpha \leq \frac{5}{3} \) and one for \( \frac{5}{3} \leq \alpha \).
Lemma 2.17. For $1 \leq \alpha \leq \frac{5}{3}$, we have that

$$\inf_{k \in \mathbb{N}} \{ A_{NF}(\alpha, k) \} \leq \frac{-\alpha^2 + 6\alpha - 1}{8\alpha}.$$ 

Proof. We will make use of Lemma 1.6. Let $d$ be an integer less than or equal to $(2 - \alpha)k$. The exact value of $d$ will be determined later on. Let $G_1$ be a $d$-regular bipartite graph with $(\alpha - 1)k + d$ vertices. The edges $E_1$ of $G_1$ are colored with the colors $C_{1,d}$. Furthermore, let $G_2$ be the complete bipartite graph $K_{k-d,k-d}$. The edges $E_2$ are colored with the colors $C_{d+1,k}$.

The adversary is going to add edges between $G_1$ and $G_2$, none of which can be colored by Next-Fit. Let $(L_1, R_1)$ and $(L_2, R_2)$ be the vertex partitioning of $G_1$ respectively $G_2$. The adversary gives the edges of a complete bipartite graph between $R_1$ and $L_2$. The vertices in $R_1$ gets degree $d + (k - d) = k$ and the vertices in $L_2$ gets degree $(\alpha - 1)k + d$ and $k$. Similarly, the adversary gives the edges of a complete bipartite graph between $R_2$ and $L_1$. These edges between $G_1$ and $G_2$ are denoted by $E_{12}$.

Since Next-Fit can only color the edges in $G_1$ and $G_2$, it colors $((\alpha - 1)k + d)k + (k - d)^2$ edges. The optimal offline algorithm OPT can choose to color $G_1$ and all the added edges $E_{12}$ between $G_1$ and $G_2$. To see why, notice that since $d \leq (2 - \alpha)k$, we have that $((\alpha - 1)k + d) \leq k$. It follows that the subgraph induced by the edges $E_1 \cup E_{12}$ is bipartite with maximum degree at most $k$.

Note that if $d$ is strictly less than $(2 - \alpha)k$, the maximum degree is also strictly less than $k$. In this case, OPT can color $k - ((\alpha - 1)k + d) = (2 - \alpha)k - d$ edges in $G_2$ for each of the $k - d$ vertices in $L_2$. Thus, the ratio of edges colored by Next-Fit to the edges colored by OPT is

$$\frac{((\alpha - 1)k + d)k + (k - d)^2}{((\alpha - 1)k + d)k + 2(k-d)((\alpha - 1)k + d) + ((2 - \alpha)k - d)(k - d)}.$$ 

If we allow $d$ to be any positive real number, we get that this ratio attain its minimum at $d = \frac{1}{4}(3 - \alpha)k$. Recall that we required $d \leq (2 - \alpha)k$. Thus, for this value of $d$ to be legit we must have that

$$\frac{1}{4}(3 - \alpha)k \leq (2 - \alpha)k.$$ 

This inequality holds since $1 \leq \alpha \leq \frac{5}{3}$. If we assume that $d = \frac{1}{3}(3 - \alpha)k$ is an integer, we get the desired upper bound for the competitive ratio

$$\frac{-\alpha^2 + 6\alpha - 1}{8\alpha}.$$ 

$\Box$
In the case where $\frac{5}{3} \leq \alpha$, the construction and analysis from [FN00] can be used almost directly. The adversary graph is similar to the one used above, but in this case we do not need to have $\text{OPT}$ being able to color some edges of $G_2$.

**Lemma 2.18.** For $\frac{5}{3} \leq \alpha \leq \sqrt{3}$, we have that

$$\inf_{k \in \mathbb{N}} \{A_{\text{Next-Fit}}(\alpha, k)\} \leq \frac{\alpha^2 - 3\alpha + 3}{\alpha}.$$  

If $\sqrt{3} \leq \alpha$, then

$$\inf_{k \in \mathbb{N}} \{A_{\text{Next-Fit}}(\alpha, k)\} \leq 2\sqrt{3} - 3.$$  

**Proof.** The adversary graph is similar to the one used in the proof of Lemma 2.17. Again, we will make use Lemma 1.6. Let $d$ be an integer less than $k$. The exact value of $d$ will be determined later on. Let $G_1 = (L_1 \cup R_1, E_1)$ be a $d$-regular bipartite graph with $k$ vertices in each vertex partition. Let $G_2 = (L_2 \cup R_2, E_2)$ be the complete bipartite graph $K_{k-d,k-d}$. The edges $E_1$ of $G_1$ is colored with the colors $C_{d+1,k}$. The adversary gives the edges of a complete bipartite graph between $R_1$ and $L_2$ and also between $R_2$ and $L_1$. Denote these edges by $E_{12}$. The edges $E_{12}$ cannot be colored by $\text{Next-Fit}$. On the other hand, the subgraph induced by the edges $E_1 \cup E_{12}$ is bipartite and has maximum degree $k$. Thus, $\text{OPT}$ can color $E_1 \cup E_{12}$. It follows that the ratio of edges colored by $\text{Next-Fit}$ to the edges colored by $\text{OPT}$ is

$$\frac{|E_1| + |E_2|}{|E_1| + |E_{12}|} = \frac{kd + (k-d)^2}{kd + 2k(k-d)}.$$  

(2.18.1)

If we allow $d$ to be any real number, we see that this ratio attains its minimum value of $2\sqrt{3} - 3$ when $d = (2 - \sqrt{3})k$. Note however that the degree of the vertices in $L_2$ and $R_2$ is $k + (k - d) = 2k - d$ and that all other vertices had degree $k$. Thus, in order for the graph to be $\alpha k$-colorable we must make sure that $2k - d \leq \alpha k$. If $\sqrt{3} \leq \alpha$ we let $d = (2 - \sqrt{3})k$ and get the upper bound of $2\sqrt{3} - 3$. If $\frac{5}{3} \leq \alpha \leq \sqrt{3}$ we let $d = (2 - \alpha)k$, and the ratio (2.18.1) then becomes

$$\frac{\alpha^2 - 3\alpha + 3}{\alpha}.$$  

We will now show that the upper bound is actually tight by providing the corresponding lower bound. The calculations needed are a bit lengthy. Essentially, what the proof shows is that the adversary graphs used above are the worst possible. This is done by considering how bad things can locally be for any fair algorithm. That is, we fix a rejected edge and consider the worst possible configuration of the adjacent colored and rejected edges.
Theorem 2.19. Let $\mathcal{A}$ be a fair algorithm. For $1 \leq \alpha \leq \frac{5}{3}$, we have that

$$\mathcal{A}_\mathcal{A}(\alpha, k) \geq \frac{-\alpha^2 + 6\alpha - 1}{8\alpha},$$

and for $\frac{5}{3} \leq \alpha \leq \sqrt{3}$, we have that

$$\mathcal{A}_\mathcal{A}(\alpha, k) \geq \frac{\alpha^2 - 3\alpha + 3}{\alpha}.$$

Finally, if $\sqrt{3} \leq \alpha$ then

$$\mathcal{A}_\mathcal{A}(\alpha, k) \geq 2\sqrt{3} - 3 \approx 0.4641.$$

Proof. Let $G = (V, E)$ be an $\alpha k$-colorable graph. Suppose that the online algorithm $\mathcal{A}$ has been given the edges of $G$ (in some sequence). We define $E_c$ and the corresponding function $d_c$ as in Definition 2.4. Consider now a max-coloring of $G$ by OPT. Let $E_u$ to be those edges colored by OPT but not colored by $\mathcal{A}$ and let $E_d$ be those edges colored by both OPT and $\mathcal{A}$. We let $d_u(x)$ be the number of edges in $E_u$ incident to $x \in V$ and defines $d_d(x)$ similarly. Furthermore, let $m(x) = \frac{1}{2}(d_c(x) - C d_d(x))$ and let $m_u(x) = \frac{m(x)}{d_u(x)}$ if $d_u(x) \geq 1$.

The idea is the following: Suppose that whenever $\mathcal{A}$ colors an edge $e = (x, y)$, it earns a dollar. If we can find a price $0 < C < \frac{1}{2}$ such that $\mathcal{A}$ can buy all edges colored by OPT paying $C$ for each, then $\mathcal{A}$ must be $C$-competitive.

In order to determine the largest possible value of $C$, we will use the following strategy. The algorithm $\mathcal{A}$ starts by buying all the shared edges $E_d$ paying $C$ for each. Once this has been done, we distribute the remaining value to the edges in $E_u$. This is done by first giving a value of $m(x)$ to each vertex $x$. We then distribute this value equally to the edges in $E_u$ incident to $x$ so that they all gets a value of $m_u(x)$ from $x$ (assuming $d_u(x) \geq 1$).

Suppose now that we can prove that each edge in $E_u$ receives a value of at least $C$. That is, suppose that $m_u(x) + m_u(y) \geq C$ for all $(x, y) \in E_u$. Then

$$C |E_u| \leq \sum_{(x,y) \in E_u} (m_u(x) + m_u(y)) \leq \sum_{(x,y) \in E_u} (m_u(x) + m_u(y)) + \sum_{x \in V \mid d_u(x) = 0} m(x) = \sum_{x \in V} m(x) = |E_c| - C |E_d|.$$
THE ACCOMMODATING FUNCTION FOR $\alpha > 1$

We wish to determine the largest possible value of $C$ satisfying $m_u(x) + m_u(y) \geq C$ for all $(x, y) \in E_u$. First, note that we must have $d_d(x) + d_u(x) \leq k$ for all $x \in V$ since $OPT$ only has $k$ colors available. For any edge $(x, y) \in E_u$ we therefore get that

$$m_u(x) + m_u(y) = \frac{1}{2} \left( \frac{d_c(x) - Cd_d(x)}{d_u(x)} + \frac{d_c(y) - Cd_d(y)}{d_u(y)} \right) \geq \frac{1}{2} \left( \frac{d_c(x) - Cd_d(x)}{k - d_d(x)} + \frac{d_c(y) - Cd_d(y)}{k - d_d(y)} \right).$$

Now, let $m_x = \frac{d_c(x) - Cd_d(x)}{k - d_d(x)}$ and let $m_y = \frac{d_c(y) - Cd_d(y)}{k - d_d(y)}$. Since $F$ is fair, for each $(x, y) \in E_u$ we have $d_c(x) + d_c(y) \geq k$. Thus, we may assume $d_c(y) \geq \frac{1}{2}k$ without loss of generality.

By considering the derivative of $m_x$ with respect to $d_d(x)$ one sees that $m_x$ is decreasing in $d_d(x)$ if $d_c(x) \geq kC$ and increasing in $d_d(x)$ if $d_c(x) > kC$. Similarly, since we assumed $d_c(y) \geq \frac{1}{2} > kC$, the function $m_y$ is increasing in $d_d(y)$. It follows that if $d_c(x) > kC$, then

$$m_u(x) + m_u(y) \geq \frac{1}{2} \left( \frac{d_c(x)}{k} + \frac{d_c(y)}{k} \right) \geq \frac{1}{2},$$

where the last inequality holds since $F$ is fair. In this case, we see that any value of $C$ less than $\frac{1}{2}$ will do. Thus, we only need to consider the case where $d_c(x) \leq kC$. We split the analysis into two subcases depending on the value of $\alpha$.

**Case 1:** We have that $1 \leq \alpha \leq \frac{5}{3}$.

First, note that $m_u(y)$ is smallest possible when $d_u(y) + d_d(y) = k$ (as opposed to being strictly smaller than $k$). Thus, we may assume that $d_u(y) + d_d(y) = k$. Since $d(y) \leq \alpha k$, we must have $d_u(y) \leq \alpha k - d_c(y)$. This in turn gives

$$d_d(y) = k - d_u(y) \geq k - (\alpha k - d_c(y)) = (1 - \alpha)k + d_c(y).$$

If $(\alpha - 1)k \leq d_c(y)$, then $(1 - \alpha)k + d_c(y) \geq 0$ and we get a (non-trivial) lower bound on $d_d(y)$.

Recall that we are considering the case where $d_c(x) \leq kC$. Since $m_x$ is increasing in $d_d(x)$, we get a lower bound on $m_x$ by letting $d_d(x) = d_c(x)$. Conversely, since $d_c(y) > kC$, we get a lower bound on $m_y$ by letting $d_d(y)$ be as small as possible. Finally, since $d_c(x) + d_c(y) \geq k$, we get a valid lower bound on $m_x + m_y$ by setting $d_c(y) = k - d_c(x)$. Doing all of this, we get that
\[m_u(x) + m_u(y) \geq \frac{1}{2} \left( \frac{d_c(x) - C d_c(x)}{k - d_c(x)} + \frac{d_c(y) - C((1 - \alpha)k + d_c(y))}{k - ((1 - \alpha)k + d_c(y))} \right)\]

\[= \frac{1}{2} \left( \frac{d_c(x) - C d_c(x)}{k - d_c(x)} + \frac{(1 - C)d_c(y) + C(\alpha - 1)k}{\alpha k - d_c(y)} \right)\]

\[\geq \frac{1}{2} \left( \frac{d_c(x) - C d_c(x)}{k - d_c(x)} + \frac{(1 - C)(k - d_c(x)) + C(\alpha - 1)k}{\alpha k - (k - d_c(x))} \right).\]

Now, this last expression is larger than or equal to \(C\) if and only if

\[
\frac{(\alpha - 3)kd_c(x) + 2d_c(x)^2 + k^2}{\alpha k^2} \geq C.
\]

If we choose \(C\) less than the minimum value of this expression, we get a lower bound on the accommodating function. The expression attain its minimum at \(d_c(x) = -\frac{5}{4}(\alpha - 3)k\). Recall that we assumed that \((\alpha - 1)k < d_c(y)\). This in turn gives us the following inequality

\[(\alpha - 1)k < k + \frac{1}{4}(\alpha - 3)k,
\]

which is satisfied since \(\alpha \leq \frac{5}{4}\). Thus, it is indeed possible for \(d_c(x)\) to be equal to \(-\frac{5}{4}(\alpha - 3)k\) and for this value, we get the desired lower bound.

**Case 2:** We have that \(\frac{5}{3} < \alpha\). In this case, \(d_c(y)\) can be zero even when \(\text{OPT}\) colors \(k\) edges at \(y\). Thus, all we can say is that

\[m_u(x) + m_u(y) \geq \frac{1}{2} \left( \frac{d_c(x) - C d_c(x)}{k - d_c(x)} + \frac{d_c(y)}{k} \right)\]

\[\geq \frac{1}{2} \left( \frac{d_c(x) - C d_c(x)}{k - d_c(x)} + \frac{k - d_c(x)}{k} \right)\].

Where the last inequality follows from the fact that \(d_c(x) + d_c(y) \geq k\). The last expression is greater than or equal to \(C\) if and only if

\[
\frac{k^2 + d_c(x)^2 - kd_c(x)}{2k^2 - kd_c(x)} \geq C. \tag{2.19.1}
\]

If we allow \(d_c(x)\) to be any real number from 0 to \(k\), we see that left-hand side of (2.19.1) attains its minimum of \(2\sqrt{3} - 3\) when \(d_c(x) = (2 - \sqrt{3})k\). However, since \(d_c(y) \geq k - d_c(x)\) and since \(\text{OPT}\) is assumed to have \(k\) colored edges at \(y\), the degree of the vertex \(y\) is at least \(k + d_c(y) \geq k + d_c(y) \geq k + k - d_c(x) = 2k - d_c(x)\). Thus, in order to obtain a valid lower bound we must have \(2k - d_c(x) \leq \alpha k\) which is equivalent to

\[(2 - \alpha)k \leq d_c(x).\]
If $\alpha \geq \sqrt{3}$, then we can choose $d_c(x) = (2 - \sqrt{3})k$. This gives the lower bound of $2\sqrt{3} - 3$ when $\alpha \geq \sqrt{3}$. If $\frac{5}{3} < \alpha < \sqrt{3}$, the worst that can happen is that $d_c(x) = (2 - \alpha)k$. In this case, the expression \((2.19.1)\) attains a value of \(\frac{\alpha^2 - 3\alpha + 3}{\alpha}\).

\[ \text{Competitive ratio for } k = 2 \]

The analysis of the accommodating function for general values of $k$ did not give any separation results. That is, we do not know of any algorithm (not even a randomized and/or unfair algorithm) which is provably significantly better than the worst possible fair algorithm \textit{Next-Fit} for all values of $k$. What we do know is that \textit{First-Fit} has a better competitive ratio for small values of $k$ when $\alpha \leq 1$. This seems to indicate that it is interesting to study the problem for small values of $k$.

For this section, we let $k = 2$ and consider the competitive ratio of the max 2-coloring problem without any restrictions on the input graph. We already know that any fair algorithm has a competitive ratio of at least $\frac{1}{2}$ for $k = 2$ (see Theorem 2.13). On the other hand, we also know that any fair deterministic algorithm has a competitive ratio of at most $\frac{1}{2}$ for all values of $k$ and hence also for $k = 2$. This result can be strengthened as follows.

\textbf{Lemma 2.20. Assume that } \mathcal{A} \text{ is fair or deterministic (or both). Then the competitive ratio of } \mathcal{A} \text{ is } \frac{1}{2} \text{ for } k = 2. \]

\textbf{Proof.} If $\mathcal{A}$ is fair, the adversary gives the edges of a 4-cycle. The edges are given so that each new edge except the first is adjacent to at least one of the edges already given. Since $\mathcal{A}$ is fair, it must color all edges in this cycle in such a way that both colors are present at all four vertices. The adversary then gives two additional edges at each of the vertices in the 4-cycle, none of which can be colored by $\mathcal{A}$. On the other hand, OPT can reject the edges in the 4-cycle and color all the other edges. See figure 2.5 on the following page. This gives a competitive ratio of $\frac{1}{2}$.

Assume now that $\mathcal{A}$ is unfair but deterministic. The adversary will give a path of a certain length depending on $\mathcal{A}$. The path is given so that each new revealed edge is connected to the already existing path. Let $N$ be the number of edges that $\mathcal{A}$ will color before it rejects the next edge in the path (note that if $N \leq 1$, then $\mathcal{A}$ is clearly at most $\frac{1}{2}$-competitive).

Assume first that $N$ is unbounded, meaning that $\mathcal{A}$ will color the entire path no matter how long it is. When the adversary has given $n$ edges that forms a path, it may give two edges at each internal vertex on the path. The algorithm $\mathcal{A}$ will not be able to color any of these additional edges, but OPT could have rejected all edges in the path and colored all other edges. This gives
COMPETITIVE RATIO FOR $K = 2$

Figure 2.5: Any fair algorithm will color the 4-cycle and reject all the dashed edges. $\text{OPT}$ will reject all edges in the 4-cycle and color all the dashed edges. This gives a ratio of $\frac{4}{8} = \frac{1}{2}$.

a competitive ratio of $\frac{n}{2(n-1)}$ which tends to $\frac{1}{2}$ as $n$ tends to infinity. Since we are assuming that there is no upper bound on $n$, this gives the desired result.

The other case we need to consider is when $1 < N < \infty$. After having colored $N$ edges, $A$ rejects the next edge $e'$ in the path. The adversary will then let $e'$ be a leaf edge in the path and continue to give edges connected to the other end of the path. If $A$ does not reject any more edges, the analysis of the previous case still applies. Assume therefore that at some point, $A$ again rejects an edge $e''$. The path now consists of $n$ colored edges and two rejected leaf edges $e'$ and $e''$. The adversary then gives two edges connected to each internal vertex in the colored part of the path (that is, to each vertex in the path which is not the endpoint of $e'$ or $e''$). Clearly, $A$ cannot color any of these additional edges. On the other hand, $\text{OPT}$ may color the edges $e'$ and $e''$ and reject all other edges from the initial path. It will then be able to color all the additional edges, giving a ratio of $\frac{n}{2(n-1)+2} = \frac{1}{2}$.

Figure 2.6: An example of the adversary graph used when $A$ might be unfair. Here, $A$ colored four edges of the path and rejected the two outmost edges. This gives a ratio of $\frac{4}{8} = \frac{1}{2}$.

What is left to consider is if a randomized unfair algorithm might achieve a better competitive ratio. Judging from the adversary graphs we have already seen, the algorithm must have a certain probability of rejecting an edge preventing the algorithm from always coloring all edges in an even cycle or on a “long” path. We state this as an open problem.

**Problem 2.21.** For $k = 2$, either find an unfair randomized algorithm with a competitive ratio better than $\frac{1}{2}$ or improve on the upper bound of $\frac{4}{7}$. 
COMPETITIVE RATIO FOR $K = 2$

The same problem is of course open for any $k \geq 2$. However, the special case of $k = 2$ seems tractable. In particular, for $k = 2$ we give a complete solution to the problem when the graphs are restricted to be 2-colorable in Theorem 3.6.
3 Paths and Trees

In this chapter, we will consider the max-coloring problem on paths and trees. Unless otherwise mentioned, we will always consider the case where the path or tree is $k$-colorable. This may seem very restrictive, but see Theorem 3.11 and the discussion following it. The main results are:

- On paths, First-Fit is optimal among deterministic algorithms with a competitive ratio of $\frac{2}{3}$. The competitive ratio of Next-Fit is $\frac{1}{2}$. Furthermore, there exists an randomized algorithm with competitive ratio of $\frac{1}{3}$ (against an oblivious adversary). We show that this randomized algorithm is in fact optimal.

- On trees, the competitive ratio of any fair algorithm tends to 1 as $k$ tends to infinity. Nevertheless, First-Fit seems to be better than Next-Fit on trees. In particular, on $k$-colorable trees First-Fit always colors at least $\Theta(k)$ edges per rejection whereas Next-Fit colors only $\Theta(\sqrt{k})$ edges pr. rejection in the worst-case. First-Fit is shown to be almost optimal among deterministic algorithms. We also show Rand cannot be much better than First-Fit.

3.1 Paths

We will consider the problem of coloring paths online with two colors available. The results obtained are summarized in Figure 3.1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fair algorithm</td>
<td>$\frac{1}{2}$</td>
<td>-</td>
</tr>
<tr>
<td>Deterministic algorithm</td>
<td>-</td>
<td>$\frac{2}{3}$</td>
</tr>
<tr>
<td>Any algorithm</td>
<td>-</td>
<td>$\frac{4}{5}$</td>
</tr>
<tr>
<td>Next-Fit</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>First-Fit</td>
<td>$\frac{2}{3}$</td>
<td>$\frac{2}{3}$</td>
</tr>
<tr>
<td>Rand</td>
<td>$\frac{3}{4}$</td>
<td>$\frac{3}{4}$</td>
</tr>
<tr>
<td>$\mathbb{E}_p$ for $p = \frac{\sqrt{\phi}}{\sqrt{5}}$</td>
<td>$\frac{4}{5}$</td>
<td>$\frac{4}{5}$</td>
</tr>
</tbody>
</table>

Figure 3.1: Lower and upper bounds for the competitive ratio of different algorithms on paths
Note that for $k \geq 3$, any fair algorithm would be 1-competitive on paths. Also, for $k = 1$ any fair algorithm would have a competitive ratio of $\frac{1}{2}$. This is the reason for only considering the case where $k = 2$.

We already know that any fair deterministic algorithm will be at least $\frac{1}{2}$ competitive. To see this directly, simply note that any rejected edge must be adjacent to two colored edges. The following result shows that $\text{Next-Fit}$ is worst possible among fair algorithms on paths.

**Theorem 3.1.** The algorithm $\text{Next-Fit}$ has a competitive ratio of exactly $\frac{1}{2}$ on paths.

**Proof.** The adversary always maintains a connected path denoted the main path. At the beginning, the main path consists of a single edge colored with the color 1. The main path is extended by repeating the following two steps:

In the first step, an edge $e$ which is not connected to the main path is given. In the second step, an edge connecting the main path to the edge $e$ is given. After step two we always have a connected main path and may repeat the procedure. Since $\text{Next-Fit}$ will alternately color the edge $e$ from the first step with the colors 1 and 2, it will always reject the edge from step two. This gives a competitive ratio of $\frac{1}{2}$. See figure 3.2

\[1 \rightarrow \ldots \rightarrow 2 \rightarrow \ldots \rightarrow 1\]

Figure 3.2: $\text{Next-Fit}$ has a competitive ratio of $\frac{1}{2}$ on paths.

**Theorem 3.2.** A deterministic algorithm has a competitive ratio of at most $\frac{2}{3}$ on paths.

**Proof.** Consider first the case of a fair, deterministic algorithm. The adversary constructs the path as follows (see also Figure 3.3 on the following page). It always maintains a connected path where one of the leaf edges is colored 2. We will refer to this path as the main path and the edge colored 2 as the rightmost edge. At the beginning, the main path will consist of two adjacent edges. The path is then extended by repeating the following two steps: In the first step, two adjacent edges (which are not connected to the main path) are given. Since we assume our algorithm is fair, it colors these two edges using the colors 1 and 2. In the second step, the adversary gives an edge connecting the edge from the first step colored 1 to the rightmost edge of the main path. This edge cannot be colored by the algorithm since the rightmost edge of the main path is colored 2. Notice that the rightmost edge of the main path is still colored 2 after step two. This allows the adversary to repeat the steps above. In each iteration, the algorithm colors two edges and rejects one. Since the adversary can make the path arbitrarily long, this implies that the competitive ratio is at most $\frac{2}{3}$ for any fair deterministic algorithm.
A RANDOMIZED ALGORITHM

The upper bound of $\frac{2}{3}$ also holds for unfair deterministic algorithms. If, at any point, an unfair algorithm rejects one of the two new edges in step one, the adversary simply does not connect these edges to the main path and repeats step one.

$$1 \rightarrow \ldots \rightarrow 2 \rightarrow 1 \rightarrow \ldots \rightarrow 2 \rightarrow 1$$

Figure 3.3: No deterministic algorithm has a competitive ratio better than $\frac{2}{3}$ on paths.

The above upper bound is tight. Indeed, the algorithm First-Fit has a ratio of exactly $\frac{2}{3}$ on paths.

**Corollary 3.3.** First-Fit is best possible among deterministic algorithms on paths.

*Proof.* The competitive ratio of First-Fit on general 2-colorable graphs is $\frac{2}{3}$. This gives a lower bound of $\frac{2}{3}$ for the competitive ratio of First-Fit on paths. Combining this with Theorem 3.2 gives the desired result.

Our upper bound in Theorem 3.2 depends crucially on the algorithm at consideration to be deterministic. The way in which the adversary gives the edges depends on how the algorithm decides to color them. In particular, the construction depends on which of the two added edges gets the color 1.

A randomized algorithm

In this section, we show that there is a randomized algorithm achieving a competitive ratio of $\frac{4}{5}$ and that this is best possible.

**Definition 3.4.** For $0 \leq p \leq 1$, let $R_p$ be the following fair randomized algorithm: Whenever an edge may be colored with either of the colors 1 or 2, the algorithm $R_p$ uses color 1 with probability $p$ and color 2 with probability $1 - p$.

Note that $R_{\frac{1}{2}}$ is identical to the general algorithm Rand restricted to the case $k = 2$. Also, $R_1$ (and $R_0$) is identical to First-Fit.

**Theorem 3.5.** The competitive ratio of $R_p$ against an oblivious adversary on paths is

$$\min \left\{ p^2 - p + 1, \frac{1}{3}(-2p^2 + 2p + 2) \right\}.$$

In particular, the best possible competitive ratio of $R_p$ is $\frac{4}{5}$ which is obtained by setting $p = \frac{1}{2} + \frac{1}{2\sqrt{3}} \approx 0.72$. 

Proof. Fix $0 \leq p \leq 1$. We will first prove the lower bound. The corresponding upper bound will follow easily from this.

Let $P$ be a (collection of) paths and let $\sigma$ be the sequence in which the edges of $P$ arrives. We need to determine $\frac{\mathbb{E}[R_p(\sigma)]}{\text{OPT}(\sigma)}$ where the expectation is over the random choices of $R_p$. Consider an edge $e \in \sigma$ at the time of its arrival. If both of the two possible adjacent edges of $e$ has already been given, we say that $e$ is a critical edge. Denote by $E_{\text{critical}}$ the critical edges of $P$. Note that since $R_p$ is fair, it will never reject an edge which is not critical. Using linearity of expectation, we get that

$$E[R_p(\sigma)] = \sum_{e \in E_{\text{critical}}} P(e \text{ is colored}) + \sum_{e \notin E_{\text{critical}}} 1. \quad (3.5.1)$$

Assume that $e$ is an edge which is not critical. Consider the largest connected component $P_e$ induced by edges from $E \setminus E_{\text{critical}}$ containing $e$. Let $e_1$ be the edge in $P_e$ which arrived first. We define $l(e)$ to be the length of the shortest path in $P_e$ containing $e$ and $e_1$. If $e$ is given as an isolated edge, then $l(e) = 1$. We say that $e$ is odd if $l(e)$ is odd and that $e$ is even if $l(e)$ is even.

Claim: If $l(e)$ is odd, then the probability of $e$ being colored with the color 1 is $p$. If $l(e)$ is even, then the probability of $e$ being colored with the color 1 is $1 - p$. We will now prove the claim. If $l(e) = 1$, then by definition of $R_p$, the edge $e$ receives the color 1 with probability $p$. If $l(e) > 1$, then consider the edge $e_1$ as defined above. The probability of $e_1$ being colored with the color 1 is $p$. Since there are no critical edges on the path from $e$ to $e_1$, we know that the edges between $e_1$ and $e$ must have been given so that they are always adjacent to exactly one other edge on the path from $e$ to $e_1$ at the time of arrival. Thus, the coloring of $e_1$ uniquely determines the coloring of all other edges on the path from $e_1$ to $e$. Namely, if $l(e)$ is odd, then $e$ is colored the same as $e_1$ and if $l(e)$ is even, then $e$ is colored the opposite of $e_1$. This proves the claim.

Let $e_c$ be a critical edge. Denote by $e_l$ and $e_r$ the two non-critical edges adjacent to $e_c$. We consider three different cases:

Case 1: $e_l$ is odd and $e_r$ is odd. In this case, the probability of $e_c$ being colored is $p^2 + (1 - p)^2$ since this happens when $e_l$ and $e_r$ are colored with the same colors. Since $e_l$ and $e_r$ are not critical, they both contribute with a value of one to the sum in (3.5.1). We will take a value of $\frac{1}{2}$ from each of them and associate to the critical edge $e_c$ (formally this can be done by rearranging the sum). The edge $e_c$ contributes with a value of $p^2 + (1 - p)^2$ to the sum. Thus, together with the associated values from $e_l$ and $e_r$ we get a value of $p^2 + (1 - p)^2 + 1 = 2(p^2 + 1 - p)$. On the other hand, $\text{OPT}$ gets a value of 2 from the same edges. This gives a ratio of

$$\frac{2(p^2 + 1 - p)}{2} = p^2 - p + 1.$$
Case 2: \( e_l \) is odd and \( e_r \) is even. In this case, the probability of \( e_c \) being colored is \( p(1-p) + (1-p)p \). Since \( e_r \) is even, it must be adjacent to at least one non-critical edge \( e'_r \). We can associate a value of \( \frac{1}{2} \) from \( e_l \) and \( e'_r \) and a value of 1 from \( e_r \) to \( e_c \). We can do this since \( e'_r \) is non-critical and hence \( e_r \) cannot be adjacent to another rejected edge besides \( e_c \). \( \text{OPT} \) gets a value of 3 from the same edges. This gives a ratio of

\[
\frac{p(1-p)(1-p)p + 2}{3} = \frac{1}{3}(-2p^2 + 2p + 2)
\]

Case 3: \( e_l \) is even and \( e_r \) is even. In this case, the probability of \( e_c \) being colored is \((1-p)^2 + p^2 \). Thus, in this case \( \text{OPT} \) gets a ratio at least as good as the ratio from case 1 (it is actually better since we can associate a value of 1 from each of \( e_l \) and \( e_r \)).

This finished the proof of the lower bound. For the upper bound, the adversary can look at the value of \( p \) (which is known even to an oblivious adversary). Depending on whether \( p^2 - p + 1 \) or \( \frac{1}{3}(-2p^2 + 2p + 2) \) is smallest, the adversary can make a path where the analysis from case 1 or case 2 is tight. This can be done by making sure that whenever \( e \) is odd then \( l(e) = 1 \) and whenever \( e \) is even then \( l(e) = 2 \). This is sufficient to make the analysis tight because it will ensure that any odd edge is adjacent to two critical edges and that any even edge is adjacent to one critical edge.

We conclude that the best possible ratio is achieved when \( p^2 - p + 1 = \frac{1}{3}(-2p^2 + 2p + 2) \). This happens when \( p = \frac{1}{2} + \frac{1}{2\sqrt{5}} = \frac{\phi}{\sqrt{5}} \) where \( \phi \) is the golden ratio (we also have equality when \( p = \frac{1}{2} - \frac{1}{2\sqrt{5}} \), but by symmetry these two values of \( p \) gives the same algorithm). When \( p = \frac{\phi}{\sqrt{5}} \), the competitive ratio of \( \text{R}_p \) is \( \frac{4}{5} \).

It follows from Theorem 3.5 that the competitive ratio of \( \text{Rand} \) on paths is \( \frac{3}{4} \). This is worse than the optimal ratio of \( \text{R}_p \) but still better than any deterministic algorithm. The algorithm \( \text{Rand} \) has the advantage of requiring only a single random bit whenever it has to choose which color to use.

We will now show that \( \text{R}_p \) for \( p = \frac{\phi}{\sqrt{5}} \) is in fact optimal among all algorithms on paths.

**Theorem 3.6.** The competitive ratio of any algorithm against an oblivious adversary on paths is at most \( \frac{4}{5} \) when \( k = 2 \). In particular, \( \text{R}_p \) is optimal when \( p = \frac{\phi}{\sqrt{5}} \).

**Proof.** We have already seen in Theorem 3.5 that \( \text{R}_p \) has a competitive ratio of at least \( \frac{4}{5} \) when \( p = \frac{\phi}{\sqrt{5}} \). Thus, we only need to give an upper bound. This will be done using Yao’s minimax principle.

Fix some \( n \in \mathbb{N} \) such that \( n = 3^{\log_3(n)} \). We will describe a probability distribution over all the possible sequences in which the edges in a path of length \( n - 2 \) can arrive. The adversary will give the edges in a number of
phases. The maximum number of phases will be $\log_3(n)$ and the minimum number of phases will be 1. Phase $i$ begins with the adversary giving $\frac{1}{3^i}n$ isolated edges. These edges are called the initial edges of phase $i$. After having given these initial edges of phase $i$ where $i < \log_3(n)$, one of two things can happen: With probability $\frac{1}{2}$ the adversary will go into a finalization state (where it will make no more phases and instead give all remaining edges of the path in a certain way) and with probability $\frac{1}{2}$ it will go into continuation state (where it will give the initial edges of the next phase). We will now give a precise description of these two possible states. What the adversary does if it ever gets to give the initial edges of phase $i = \log_3(n)$ will be described separately later on.

With probability $\frac{1}{2}$, the adversary goes into continuation state. In this case, it will go on to give the $\frac{1}{3^{i+1}}n$ initial edges of phase $i + 1$.

With probability $\frac{1}{2}$, the adversary goes into finalization state. It will connect the initial edges of phase $i$ using two edges per connection. That is, the adversary first gives an edge $e'$ adjacent to exactly one of the initial edges and then another edge $e''$ adjacent to $e'$ and exactly one of the initial edges. This will result in $\frac{3}{3^i}n - 2$ edges being given in phase $i$ (including the initial edges). The adversary then connects all initial edges of phase 1 to $i - 1$ (if $i = 1$, it does nothing) using a single edge for each connection. It also connects one of the initial edges from phase $i - 1$ to one of the initial edges of phase $i$. In total, this will result in $n \sum_{p=1}^{i-1} 2^{\frac{1}{3^p}} = n - \frac{3}{3^i}n$ edges being given where we count both initial edges and connecting edges of phase 1 to $i - 1$. Together with the $\frac{2}{3^i}n - 2$ from phase $i$, this gives $n - 2$ edges and the path is therefore finished.

If the adversary ever gets to the point of giving the initial edges in phase $\log_3(n)$, it will finalize the path by connecting each of the $n \sum_{i=1}^{\log_3(n)} \frac{1}{3^i}$ initial edges of all the $\log_3(n)$ phases using a single edge for each connection. After this, the adversary will have given $2n \sum_{i=1}^{\log_3(n)} \frac{1}{3^i} - 1 = n - 2$ edges (this equality is true since $3^{\log_3(n)} = n$).

Let $D$ be any deterministic algorithm for the max 2-edge coloring problem. We will show that $E_\sigma(D(\sigma)) \leq \frac{4}{5}n$ where the expectation is over the probability distributions on sequences of edges $\sigma$ described above.

For each of the initial edges $e$, there are four different possibilities: The edge $e$ can be colored the same as the previous given initial edge, it can be colored differently from the previous given edge, it can be rejected and it can be colored while the previous given edge is rejected. For the very first edge in the phase 1, none of these may apply but we can ignore this by letting $n$ tend to infinity. Fix $1 \leq i \leq \log_3(n)$. Let $E^s_i$ be those initial edges in phase $i$ which are colored with the same color as the previous initial edge (which may be the last edge of phase $i - 1$). We define $E^d_i$ similarly, but for edges colored with a different color than the previous initial edge. Finally, let $E^r_i$ be those initial edges of phase $i$ which are rejected. We will need the following observation.
Claim: In any phase $i$, we have that $|E^*_{i}| + |E^d_{i}| + 2|E^r_{i}| \geq \frac{1}{3^i} n$. 

Since there are $\frac{1}{3^i}$ initial edges in phase $i$, the claim is clearly true if no edges are rejected. If an edge is rejected, then the next given initial edge may not be in any of the three classes. But note that this can happen at most once per rejection, and hence $|E^r_{i}|$ bounds the number of times this happens. This proves the claim.

We will now calculate the expected number of rejected edges in phase $i$ (assuming that the adversary reaches phase $i$). Note that for each edge in $E^*_{i}$, the algorithm $D$ makes a rejection with probability $\frac{1}{2}$ since it will be forced to do so if phase $i$ is the final phase of the path. Conversely, for each edge in $E^d_{i}$ the algorithm $D$ makes a rejection with probability $\frac{1}{2}$ since this forces a rejection if and only if phase $i$ is not the final phase. Also, for each edge in $E^r_{i}$ we get a rejection with probability 1. In total, the expected number of rejections in phase $i$ after the adversary has connected all the initial edges is 

$$ \frac{1}{2} |E^*_{i}| + \frac{1}{2} |E^d_{i}| + |E^r_{i}| = \frac{1}{2}(|E^*_{i}| + |E^d_{i}| + 2|E^r_{i}|) \geq \frac{1}{2} \frac{1}{3^i} n. $$

The probability that the adversary ever gets to making phase $i$ is $\frac{1}{2^{i-1}}$. Thus, the expected number of rejections over the given probability distribution is at least

$$ n \sum_{i=1}^{\log_3(n)} \frac{1}{2^{i-1}} \frac{1}{2} \frac{1}{3^i} = n \sum_{i=1}^{\log_3(n)} \frac{1}{6^i}. $$

From this, we get that the expected competitive ratio is at most

$$ \frac{E_{\sigma}(D(\sigma))}{OPT(\sigma)} \leq \frac{n \left(1 - \sum_{i=1}^{\log_3(n)} \frac{1}{6^i}\right)}{n} $$

$$ = 1 - \sum_{i=1}^{\log_3(n)} \frac{1}{6^i} $$

$$ = \frac{1}{5} \left(4 + \frac{1}{6^{\log_3(n)}}\right), $$

which tends to $\frac{4}{5}$ when $n$ tends to infinity. \qed
3.2 Trees

In this section, we will investigate how well our algorithms do on trees. The main results are the following. On trees, the competitive ratio of any fair algorithm tends to 1 as \( k \) tends to infinity. Nevertheless, First-Fit seems to be better than Next-Fit on trees. In particular, on \( k \)-colorable trees First-Fit always colors at least \( k - 2 \) edges per rejection whereas Next-Fit can be made to color only \( 2\sqrt{k} - 2 \) edges per rejection in the worst case. No deterministic algorithm can color more than \( k + 1 \) edges per rejection, and so First-Fit is close to being optimal. We also show that Rand cannot color more than \( k + 3 \) edges per rejection, which seems to imply that this simple randomized algorithm is not significantly better than First-Fit on trees.

In addition, we give a lower bound on the competitive ratio for any fair algorithm on \( k \)-regular trees and on binary trees. This was done mostly since I initially found it difficult to analyze the general case. However, these results do not follow from the general results and uses some different techniques which can be useful for gaining some intuition about the problem.

We summarize our results for the competitive ratio on \( k \)-colorable trees in the following table. The lower bound for fair algorithms also holds without the assumption that the tree is \( k \)-colorable. See also Figure 3.7 on page 43.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fair algorithm</td>
<td>( \frac{2\sqrt{k} - 2}{2\sqrt{k} - 1} )</td>
<td>( \frac{k+1}{k+2} )</td>
</tr>
<tr>
<td>Deterministic algorithm</td>
<td>( \frac{2\sqrt{k} - 2}{2\sqrt{k} - 1} )</td>
<td>( \frac{k+1}{k+2} )</td>
</tr>
<tr>
<td>Next-Fit</td>
<td>( \frac{2\sqrt{k} - 2}{2\sqrt{k} - 1} )</td>
<td>( \frac{2\sqrt{k} - 2}{2\sqrt{k} - 1} )</td>
</tr>
<tr>
<td>First-Fit</td>
<td>( \frac{k-2}{k-1} )</td>
<td>( \frac{k}{k+1} )</td>
</tr>
<tr>
<td>Rand</td>
<td>( \frac{k+3}{k+4} )</td>
<td>( \frac{k+4}{k+5} )</td>
</tr>
</tbody>
</table>

General lower and upper bounds

We will consider the problem of edge-coloring \( k \)-colorable trees (and forests) with \( k \) colors available. A tree is \( k \)-colorable if and only if the maximum degree of the tree is at most \( k \). We have already investigated the case when \( k = 2 \) since a 2-colorable tree is just a path. Our first result is a lower bound for fair algorithms.

**Theorem 3.7.** If \( F \) is a fair algorithm, then it has a competitive ratio of at least

\[
\frac{2\sqrt{k} - 2}{2\sqrt{k} - 1}
\]

on \( k \)-colorable trees.
**Proof.** Let $T$ be a $k$-colorable tree. We will view $T$ as a rooted tree. Consider the coloring produced by $F$ when given the edges of $T$ in some sequence. As in Theorem 2.6, we will think of $F$ gaining one unit of some value whenever it colors an edge. Our goal is to show that we can distribute these earnings among the rejected edges so that each of them gets a value of at least $2\sqrt{k} - 2$.

We will not use the strategy described in Theorem 2.6, but instead use the tree structure. Each vertex $y$ (except for the root) has a single parent edge and between zero and $k-1$ children edges.

(i) If the parent edge of $y$ is a rejected edge and $d_c(y) \leq 2\sqrt{k} - 2$, we give the parent edge a value of $d_c(y)$ coming from the colored children edges. If $d_c(y) > 2\sqrt{k} - 2$, we give the parent edge a value of $2\sqrt{k} - 2$ and divides the remaining value of $d_c(y) - (2\sqrt{k} - 2)$ equally among the $d_r(y) - 1$ rejected children edges (if there are any).

(ii) If the parent edge of $y$ is a colored edge, then we give a value of $\frac{d_c(y) - 1}{d_r(y)}$ to each of the $d_r(y)$ rejected children edges of $y$ (if there are any).

![Figure 3.4: The rejected edge $(x, y)$ receives a value of one unit from each of its $2\sqrt{k} - 2$ of its colored children edges (or from as many as possible if there are fewer than $2\sqrt{k} - 2$). The value from the remaining colored children edges is distributed equally among the rejected children edges.](image)

Now fix an edge $e = (x, y)$ which has been rejected by $F$. We will assume that $x$ is the parent of $y$. If $d_c(y) \geq 2\sqrt{k} - 2$, then $e$ must receive a value of at least $2\sqrt{k} - 2$ from its own colored children edges. Assume therefore that $d_c(y) < 2\sqrt{k} - 2$. Furthermore, assume that the parent edge of the vertex $x$ has been rejected (if not, we will get an even larger value at $e$). Since $F$ is fair, we must have $d_c(x) \geq k - d_c(y)$. Following the strategy described above, we must first transfer a value of $2\sqrt{k} - 2$ to the rejected parent edge of $x$. After doing this, we will have a value of $d_c(x) - (2\sqrt{k} - 2)$ left. In the worst case, there will be $d_c(y) - 1$ rejected children edges at $x$ that needs to share the remaining value. There cannot be more than this since the tree is assumed to be $k$-colorable. Combining these observations, we get that the value received
by $e$ can be lower bounded as
\[ \frac{d_c(x) - (2\sqrt{k} - 2)}{d_c(x) - 1} + d_c(y) \geq \frac{k - d_c(y) - (2\sqrt{k} - 2)}{d_c(y) - 1} + d_c(y). \]

When $1 \leq d_c(y) < 2\sqrt{k} - 2$, this expression has a minimum of $2\sqrt{k} - 2$ at $d_c(y) = \sqrt{k}$. This completes the proof.

As opposed to the general case (of $k$-colorable graphs), we see that it becomes easier to edge-color a $k$-colorable tree when $k$ grows. In particular, the lower bound implies that the competitive ratio of any fair algorithm on $k$-colorable trees tends to 1 as $k$ tends to infinity.

In the above proof, there is no explanation as to why we use a value of $2\sqrt{k} - 2$. The value was determined by first considering an arbitrary value $C$ such that
\[ \frac{k - d_c(y) - C}{d_c(y) - 1} + d_c(y) \geq C. \]
If one allows $C$ to be any real number (between 0 and $k$), the largest value of $C$ that satisfies the inequality is exactly $2\sqrt{k} - 2$. It might be possible to get a slightly better lower bound when $k$ is not a square number. In particular, for $k = 2$ we already know that the tight lower bound for fair algorithms is $\frac{1}{2}$ but $\frac{2\sqrt{2} - 2}{2\sqrt{2} - 1} \approx 0.45$. For $k = 3$, we will give a tight lower bound of $\frac{2}{3}$ in Lemma 3.16. This is slightly better than $\frac{2\sqrt{3} - 2}{2\sqrt{3} - 1} \approx 0.59$. We will not look further into the cases where $k \geq 4$ and $\sqrt{k}$ is not an integer. We will however show that when $\sqrt{k}$ is an integer, our lower bound is tight.

In order to show that the lower bound we obtained in Theorem 3.7 is tight when $k$ is a square number, we show that the competitive ratio of Next-Fit exactly matches the lower bound when $k$ is a square number.

**Theorem 3.8.** For $k \geq 4$, the algorithm Next-Fit has a competitive ratio of at most
\[ \frac{k}{\lceil \sqrt{k} \rceil} + \lceil \sqrt{k} \rceil - 2 \]
on $k$-colorable trees. In particular, if $k$ is a square number then the competitive ratio is
\[ \frac{2\sqrt{k} - 2}{2\sqrt{k} - 1}. \]

**Proof.** We will describe a $k$-colorable tree and specify a coloring of a subset of the edges in this tree. The coloring is such that none of the remaining edges in the described tree can be colored when we only have $k$ colors available. By Lemma 1.6, the adversary can force Next-Fit to produce the specified coloring by making $k$ copies of $T$. Once this has been done, the adversary gives the remaining edges to Next-Fit in any order. By construction, Next-Fit is forced to reject all of these edges.
Fix $k \geq 4$ and let $N \in \mathbb{N}$. The adversary graph consists of $N$ blocks. For $1 \leq i \leq N$, the $i$'th block contains a fan with $k - \lceil \sqrt{k} \rceil$ edges with the colors $C_{1,\lceil \sqrt{k} \rceil}$. The base vertex of this fan is denoted $v_i$. It also contains $\lfloor \sqrt{k} \rfloor - 1$ fans with $\lfloor \sqrt{k} \rfloor$ edges. These fans will be referred to as the small fans. In each of the small fans, the edges are colored with $C_{\lceil \sqrt{k} \rceil + 1, k}$. Finally, there is a single edge between $v_i$ and the base of each of the small fans. None of these $\lfloor \sqrt{k} \rfloor - 1$ edges connecting $v_i$ to the small fans can be colored because of the specified coloring of the fans.

The $i$'th block is connected to the $i+1$'th block by an edge from $v_{i+1}$ to the base vertex of one of the small fans in the $i$'th block. These edges connecting the blocks cannot be colored. Since $k \geq 4$ we must have $\lfloor \sqrt{k} \rfloor + 2 \leq k$ and so the tree is $k$-colorable. This finished the description of the adversary graph.

In each of the blocks except the very first, there are $(k - \lceil \sqrt{k} \rceil) + (\lceil \sqrt{k} \rceil - 1)\lceil \sqrt{k} \rceil$ colored edges and $\lceil \sqrt{k} \rceil$ rejected edges. In other words, the number of edges colored pr. rejected edge in each of the blocks is

$$\frac{(k - \lfloor \sqrt{k} \rfloor) + (\lceil \sqrt{k} \rceil - 1)\lfloor \sqrt{k} \rfloor}{\lceil \sqrt{k} \rceil} = \frac{k}{\lfloor \sqrt{k} \rfloor} + \lfloor \sqrt{k} \rfloor - 2.$$

By letting $N$ tend to infinity, this gives the desired upper bound on the competitive ratio.

![Figure 3.5](image)

Figure 3.5: The adversary graph used in Theorem 3.8 for $k = 9$ and $N = 3$.

For general deterministic algorithms (fair or unfair), we have the following upper bound.

**Theorem 3.9.** No deterministic algorithm has a competitive ratio better than

$$\frac{k^2 - k + 1}{k^2}$$

on $k$-colorable trees.

We note that $\frac{k}{k+1} \leq \frac{k^2 - k + 1}{k^2} \leq \frac{k+1}{k+2}$ for $k \geq 2$. 

\[\square\]
Proof. We first assume that our algorithm is fair. The adversary gives $N$ fans for some large $N$. A fan consists of $k - 1$ edges connected at a single vertex called the base. Notice that a fair algorithm must color all edges in a fan and can do so in $k$ different ways. The adversary then connects fans colored the same in groups of $k$ fans. This is done by connecting the base vertices of the $k$ fans to a single central vertex. Only one of the newly added edges can be colored (by using the single color missing at each of the fans). Clearly, the graph is $k$-colorable. Notice that we might have one group of less than $k$ fans for each of the $k$ different fan colorings. These incomplete groups can be ignored if we allow $N$ to be arbitrarily large. Thus, we get an upper bound for the competitive ratio of

$$
\frac{(k - 1)k + 1}{(k - 1)k + k} = \frac{k^2 - k + 1}{k^2}.
$$

We claim that the same upper bound holds true for any unfair deterministic algorithm. Suppose that the algorithm rejects one or more edges in a fan. As soon as this rejection happens, the adversary stops giving edges in this fan and will never connect this incomplete fan to any other fans. A fan in which at most $k - 2$ edges has been colored gives a ratio of at most $\frac{k - 2}{k - 1}$ which is lower than (3.9.1). We conclude that we cannot archive a competitive ratio better than (3.9.1) by being unfair.

Corollary 3.10. The competitive ratio of First-Fit on $k$-colorable trees is at most $\frac{k}{k + 1}$.

Proof. Just use the construction from Theorem 3.9. The fans of size $k - 1$ will be colored with the colors $C_{1,k-1}$. When we have formed a full group, we can extend the tree at any leaf. This will give us a new group with $k - 1$ rejections but with one less colored edge. Continuing like this, the competitive ratio can be arbitrarily close to

$$
\frac{((k - 1)k + 1) - 1}{((k - 1)k + k) - 1} = \frac{k}{k + 1}.
$$

Figure 3.6: A single group of similarly colored fans in the adversary graph used in Theorem 3.9 for $k = 4$
We end this section with a result showing that the lower bound from Theorem 3.7 actually holds even if we do not restrict the tree to be $k$-colorable.

**Theorem 3.11.** If $F$ is a fair algorithm, then it has a competitive ratio of at least

$$\frac{2\sqrt{k} - 2}{2\sqrt{k} - 1},$$

on trees.

**Proof.** We will use the same notation as in Theorem 2.19. In particular, we want to find the largest possible $0 \leq C \leq 1$ such that we can “buy” all edges colored by $\text{OPT}$, paying a value of $C$ for each edge. At the very beginning, we buy all edges colored by both $\text{OPT}$ and $F$ paying $C$ for each. We then distribute the remaining value in a certain way among the edges colored only by $\text{OPT}$ in the tree.

Let $T$ be a tree. We will view $T$ as a rooted tree. Consider the coloring produced by $F$ when given the edges of $T$ in some sequence. Let $y$ be a vertex in $T$ which is not the root.

(i) If the parent edge $(x, y)$ of $y$ is a rejected edge, we transfer a value of $\min\{C, d_c(y) - C d_d(y)\}$ to $(x, y)$. If $d_c(y) - C d_d(y) > C$ and $d_r(y) \geq 2$, we furthermore transfer a value of $(d_c(y) - C d_d(y) - C) a_r(y) - 1$ to each of the rejected children edges at $y$.

(ii) If the parent edge $(x, y)$ of $y$ is colored and $d_r(y) \geq 1$, we transfer a value of $(d_c(y) - C d_d(y)) a_r(y)$ to each of the rejected children edges at $y$.

If an edge is colored by both $\text{OPT}$ and $A$, it automatically gets a value of at least $C$. If it is rejected, it also gets a value of at least $C$ from its colored children edges unless $d_c(y) - C d_d(y) < C$. Note that this can only be the case if $d_r(y) = d_d(y)$ since $0 \leq C \leq 1$. The lower bound now follows from some calculations very similar to the ones in Theorem 3.7. We sketch them here. First, for the vertex $x$ one sees that it only becomes better if $d_d(x) < d_c(x)$. Assume therefore that $d_d(x) = d_c(x)$. Now, in the worst case the edge $e = (x, y)$ gets a value of at least

$$\frac{d_c(x) - C d_c(x) - C}{d_r(x) - 1} \geq \frac{k - d_c(y) - C(k - d_c(y)) - C}{d_c(y) - 1},$$

from the children edges at $x$. Together with the value from the children edges at $y$, the total value received by $e$ is

$$\frac{k - d_c(y) - C(k - d_c(y)) - C}{d_c(y) - 1} + d_c(y) - C d_c(y).$$
Now, we see that
\[ k - d_c(y) - C(k - d_c(y)) - C \leq C \]
\[ d_c(y) - 1 + d_c(y) - C d_c(y) \geq C \]
\[ \Leftrightarrow C = \frac{k - 2d_c(y) + d_c(y)^2}{k + d_c(y)^2 - d_c(y)}. \]

This last expression attains its minimum value of \( \frac{2\sqrt{k} - 2}{2\sqrt{k} - 1} \) when \( d_c(y) = \sqrt{k} \).

This theorem indicates that it is just as difficult to color \( k \)-colorable trees as it is to color trees without any restriction on the maximum degree. Because of this, we have focused on \( k \)-colorable trees. However, we do not have a proof that all lower bounds are the same for \( k \)-colorable trees and general trees. In particular, it is not clear if Theorem 3.12 is also true for general trees.

Lower bound for First-Fit

The remaining question at this point is whether there are algorithms better than Next-Fit for fixed values of \( k \). In particular, we would like to know whether First-Fit has a better competitive ratio for fixed values of \( k \) (we know this is the case when \( k = 2 \)).

**Theorem 3.12.** Let \( k \geq 3 \). The competitive ratio of First-Fit is at least \( \frac{k - 2}{k - 1} \) on \( k \)-colorable trees.

**Proof.** Let \( T \) be a \( k \)-colorable tree. We will view \( T \) as a rooted tree. When referring to an edge \((x, y)\), we always write the parent node \( x \) first. Consider the coloring produced by First-Fit when given the edges of \( T \) in some sequence. As in Theorem 2.6, we will think of First-Fit gaining one unit of some value whenever it colors an edge. Our goal is to show that we can distribute these earnings among the rejected edges so that each of them gets a value of at least \( k - 2 \). We will use a different strategy than earlier in order to account for the specific properties of a coloring produces by First-Fit.

**Strategy:** Initially, all colored edges have a value of one unit. The following rules are applied repeatedly until any further repetition would not transfer any value at all: For all edges \( e \) colored with a color different from \( k \), one of two things may happen.

1. If the parent edge of \( e \) is an edge colored with a color higher than \( c \), the edge \( e \) transfers its current value to the parent edge. If \( e' = (u, v) \) is an colored edge with a children edge colored \( k \), there will be a limit \( l(e') \) as to how much value that can be transferred to \( e' \). If at any point this limit is reached, any further value that should be transferred to \( e' \) according to the rule above instead stays at the vertex \( v \). We define \( l(e') = c(e') - 1 \) where \( c(e') \) is the color used to color the edge \( e' \).
(ii) If the parent edge of $e$ is rejected and $e$ is not colored with the color $k$, the edge $e$ transfers its current value to its parent vertex.

We denote by $m(e)$ the value obtained by the edge $e$ after applying the strategy. Also, we denote by $m(v)$ the value obtained by the vertex $v$.

Fix an edge $e = (x, y)$ colored with the color $k$. We will show how to transfer a value of at least $k - 2$ to each of the rejected edges at $x$ and $y$. This will give the desired lower bound, since a rejected edge is always adjacent to at least one edge colored with the color $k$.

**Claim 1.** If the parent edge of the vertex $v$ is rejected, then $m(v) \geq c_{\text{max}}$ where $c_{\text{max}}$ is the largest color in $C_v$.

**Proof of claim:** In order for First-Fit to color an edge $e$ with the color $c_{\text{max}}$, it must first color edges incident to $e$ with the colors $1, \ldots, c_{\text{max}} - 1$. Let $e'$ be an edge incident to $e$ colored with one of these colors. If $e'$ is a sibling edge to $e$, we get a value of 1 added to $m(v)$ since the parent edge of $e$ (and hence $e'$) is rejected and so this value is transferred to $v$. If $e'$ is a children edge to $e$, then since the color of $e$ by assumption is higher than the color of $e'$ the value of 1 from $e'$ is first transferred to $e$ and then to $v$. □

**Claim 2.** Let $e = (x, y)$ be an edge colored with the color $k$ and let

$$c_{\text{max}} = \max \left[ C_y \setminus \{k\} \right],$$
$$c' = \max \left[ \{c \in C_y : c < c_{\text{max}}\} \cup \{0\} \right].$$

Then

$$m(e) \geq (c_{\text{max}} - c')c' + |C_y|.$$  

**Proof of claim:** By construction, $e$ will always receive a total value of $|C_y|$ from its colored children edges and itself. Consider therefore the term $(c_{\text{max}} - c')c'$. By the definition of $c'$, there must be $c_{\text{max}} - c'$ edges at $y$ colored with the colors $C_{c'+1,c_{\text{max}}}$. In particular, they are all colored with a color larger than $c'$. If $c' = 0$, there is nothing to prove, so assume $c' > 0$. Note that there is no edge at $y$ colored with the color $c'$, and hence all of them must have a children edge colored $c'$. In order for First-Fit to color an edge with $c'$, that particular edge must have edges with each of the colors $C_{1,c'-1}$ as siblings or children edges. In all cases, the value of these edges (the edge colored $c'$ and its siblings and/or children edges colored $C_{1,c'-1}$) must be transferred all the way to $e$. That is, this value must be transferred to $e$ unless it is blocked because $l(e')$ is exceeded for an edge $e'$ on the way to $e$. Note that the definition of $l(e')$ is made precisely so that this does not happen. □

**Claim 3.** Let $e = (x, y)$ be an edge colored with the color $k$ and let

$$c_{\text{max}} = \max \left[ C_x \setminus \{k\} \right],$$
$$c' = \max \left[ \{c \in C_x : c < c_{\text{max}}\} \cup \{0\} \right].$$
If the parent edge $e'$ of $e$ is rejected (or if $e$ does not have a parent edge), then

$$m(x) \geq (c_{\text{max}} - d')c' + |C_x| - 1.$$  

On the other hand, if $e'$ is colored with the color $c(e')$, then

$$m(x) \geq (c_{\text{max}} - 1 - d')c' + |C_x| - 1 - c(e').$$

Proof of claim: Assume first that $e'$ is rejected. The proof of claim 2 gives directly that there must be transferred a value of at least $(c_{\text{max}} - d')c'$ to the vertex $x$. This value will not be transferred further up in the tree since $e'$ is rejected. In addition, there is a value of $|C_x| - 1$ coming from the colored edges at $x$ except for the edge $e$ itself.

Assume now that $e'$ is colored with the color $c(e')$. We will prove the claim by arguing how much $m(x)$ can be less in this situation compared to the situation when $e'$ was rejected. We lose a value of $c'$ at $x$ if $c' < c(e') \leq c_{\text{max}}$ compared to the situation where $e'$ is rejected, since the vertex $x$ does not get any value from those edges adjacent to $e'$ that are not already adjacent to $e$. Furthermore, we need to transfer a value of $l(e) = c(e') - 1$ to the edge $e'$. Finally, we loose a value of 1 at $x$ since we do get a value of one from the coloring of $e'$. Thus, in the worst case we loose a value of $c' + c(e')$ at the vertex $x$ compared to the situation where $e'$ is rejected.

We can now prove the lower bound. This is done by first considering the rejected edges at $y$ and then at $x$. For each vertex we split into two cases, one where the color $k - 1$ is present at the vertex and one where it is not.

Consider the vertex $y$. Assume that $k - 1 \notin C_y$. Let $e_r = (y, v)$ be a rejected edge at $y$. There must be an edge colored $k - 1$ at $v$. It cannot be the parent of $v$ since this is a rejected edge. By claim 1, the vertex $v$ receives a value of at least $k - 1$. This value is associated to $e_r$. Note that we did not have any of the value transferred to $e$.

Assume now that $k - 1 \in C_y$. Then $c_{\text{max}} = k - 1$ for the vertex $y$ (recall that $c_{\text{max}}$ is the highest color in $C_y \setminus \{k\}$). By claim 2, the edge $e$ receives a value of at least $(c_{\text{max}} - d')c' + |C_y| = (k - 1 - c')c'$ coming from the children edges of $y$. Recall that $c'$ is the highest color in $C_y$ such that $c' < c_{\text{max}}$. If $c' = 0$, there is nothing to prove. If not, note that there can be no more than $c'$ rejected edges at $y$ since there must be $k - c'$ colored edges. It is therefore possible to give a value of at least

$$\frac{(k - 1 - c')c'}{c'} = k - 1 - c',$$

to each of the rejected edges at $y$. By definition, $c'$ is not in $C_y$ and therefore it must be at the other endpoint of all rejected edges. By claim 1, each rejected edge $e_r = (y, v)$ gets a value of at least $c'$ from $v$. Thus, each rejected edge $e_r$
gets a value of at least \( k - 1 \). Note that there must still be at least a value of \(|C_y|\) at \( e \).

Consider the vertex \( x \). Assume first that the parent edge \( e' \) of \( x \) is rejected. If \( k - 1 \notin C_x \), then by Claim 1 any rejected edge at \( x \) except for the edge \( e' \) gets a value of \( k - 1 \). By claim 3, there is still an unused value of \(|C_x| - 1\) and together with the unused value of \(|C_y|\) from the edge \( e \), we can transfer a value of \( k - 1 \) to \( e' \).

If \( k - 1 \in C_x \), then \( c_{\text{max}} = k - 1 \). It follows from claim 3 that we get a value of at least \((k - 1 - c')c' + |C_x| - 1\) at \( x \). By the same argument used when dealing with the vertex \( y \), each rejected edge at \( x \) except for \( e' \) gets a value of \( k - 1 \) even without using the value of \(|C_x| - 1\). This unused value together with the value of \(|C_y|\) from \( e \) is then given to \( e' \). Note that \(|C_x| - 1 + |C_y| \geq k - 1 \) since First-Fit is fair.

Assume now that \( e' \) is colored. In this case, the edge \( e' \) has already been given a value of (at most) \( l(e') \) according to the strategy. We need to show that it is still possible to associate a value of at least \( k - 2 \) to each rejected edge at \( x \). If \( k - 1 \notin C_x \), then claim 1 shows that this is indeed possible. Assume therefore that \( k - 1 \in C_x \). According to claim 3, we have a value of at least \((c_{\text{max}} - 1 - c')c' + |C_x| - 1 - c(e')\) at the vertex \( x \). Note that in this case, we have already accounted for the value that the strategy requires that we transfer to \( e' \). Since \( k - 1 \in C_x \), we must have \( c_{\text{max}} = k - 1 \). Furthermore, there is an unused value of \(|C_y|\) at the edge \( e \). Finally, there is at most \( c' \) rejections at \( x \) and we may assume that \( c' > 0 \) since otherwise there is nothing to prove. Distributing the value equally among these rejected edges gives

\[
\frac{(k - 2 - c')c' + (k - 1) - c(e')}{c'} = k - 2 - c' + \frac{k - 1 - c(e')}{c'} \geq k - 2 - c'.
\]

The last inequality follows from the fact that \( c(e') \leq k - 1 \). Since the color \( c' \) is missing at \( x \), claim 1 shows that each rejected edge must get a value of at least \( c' \). Thus, each rejected edge gets a value of at least \( k - 2 \).

\[\square\]

The above lower bound is not tight, which can be seen by considering the case of \( k = 3 \) (see Lemma 3.16). However, it still shows that the number of edges colored by First-Fit per rejection in the worst-case is of the order \( \Theta(k) \) whereas for Next-Fit, the number of colored edges per rejection in the worst-case is only \( \Theta(\sqrt{k}) \). It also proves that for any fixed \( k > 4 \), the competitive ratio of First-Fit is slightly better than the competitive ratio of Next-Fit. See Figure 3.7 on the following page.

**\( k \)-regular trees**

We will consider the problem of edge-coloring \( k \)-regular trees (and forests) with \( k \) colors available. A tree is \( k \)-regular if the degree of a vertex in the tree
Figure 3.7: The competitive ratio of Next-Fit and the lower bound on the competitive ratio of First-Fit on $k$-colorable trees.

is either 1 or $k$. In other words, all internal vertices in the tree must have degree $k$. A $k$-regular tree is always $k$-colorable.

We would expect almost any fair online algorithm to perform better on $k$-regular trees than on general $k$-colorable trees (and graphs). One of the reasons is that a fair algorithm will always color all the leaf edges of a tree, since a leaf edge is adjacent to at most $k - 1$ other edges. As $k$ grows, one would expect the number of leaf edges in a $k$-regular tree to become much larger than the number of rejected edges.

The following lemma will give us a way of counting the number of leaves by considering the degree of the internal vertices.

**Lemma 3.13.** A graph $G = (V, E)$ with no isolated vertices has $|V| - 1$ edges if and only if

$$|V_i| = 2 + \sum_{i=3}^{\infty} (i - 2)|V_i|,$$

(3.13.1)

where $V_i$ is the set of vertices in the graph with degree $i$. 
Proof. The lemma follows by rearranging the equation (3.13.1):

\[ |V_i| = 2 + \sum_{i=3}^{\infty} (i - 2)|V_i| \Leftrightarrow 2 - |V_i| + \sum_{i=3}^{\infty} (i - 2)|V_i| = 0 \]
\[ \Leftrightarrow 2 + \sum_{i=1}^{\infty} (i - 2)|V_i| = 0 \]
\[ \Leftrightarrow \sum_{i=1}^{\infty} i|V_i| = 2 \sum_{i=1}^{\infty} |V_i| - 2 \]
\[ \Leftrightarrow \frac{1}{2} \sum_{i=1}^{\infty} i|V_i| = \sum_{i=1}^{\infty} |V_i| - 1 \]
\[ \Leftrightarrow |E| = |V| - 1, \]

where we used the assumption that \( |V_0| = 0 \) in the last step.

It follows that a connected graph satisfies equation (3.13.1) if and only if the graph is a tree.

We will now give a lower bound for fair, deterministic algorithms on \( k \)-regular trees which is better than the general lower bound from Theorem 3.7. We will consider our trees to be rooted (by picking an arbitrary root). The idea is to count for each rejected edge the number of colored children and then use the edges with many colored childrens (in particular the leaf edges) to make up for the edges with a low number of colored childrens.

**Theorem 3.14.** Let \( \mathcal{A} \) be a fair, deterministic algorithm. Then the competitive ratio of \( \mathcal{A} \) on \( k \)-regular trees is at least \( \frac{k-2}{k-1} \).

**Proof.** Let \( T \) be a rooted \( k \)-regular tree. Consider an edge \( e = (x, y) \) (where \( x \) is the parent of \( y \)) rejected by \( \mathcal{A} \). We say that \( e \) is *good* if \( d_c(y) = k - 1 \) and that \( e \) is *bad* if \( d_c(y) < k - 1 \). Note that all leaf edges are good.

At first, the edge \( e \) gets a value of \( d_c(y) \). Here \( d_c(y) \) is the number of colored children edges that \( e \) has. We will describe a strategy for distributing the values associated to the edges such that all rejected edges gets a value of at least \( k - 2 \). There are two cases to consider, depending on whether \( e \) is good or bad.

If \( e \) is bad, then it has \( d_c(y) \) colored children edges and \( d_r(y) - 1 \) rejected children edges. Consider now the subgraph \( T_r \) induced by the rejected edges in \( T \). Using Lemma 3.13, we know that the \( d_r(y) \) rejected edges at \( y \) can be associated with \( d_r(y) - 2 \) leaf edges of \( T_r \). We now transfer a value of 1 from each of these leaf edges to the edge \( e = (x, y) \). Since the tree is \( k \)-regular we know that \( d_c(y) + d_r(y) = k \) and so the edge \( e \) receives a total value of

\[ d_c(y) + (d_r(y) - 2) = k - 2. \]
If \( e \) is good, it gets a value of \( k - 1 \). Since any good edge is associated to at most one bad edge in the tree, the good edge will give away at most a value of 1. In total, all rejected edges will end up with a value of at least \( k - 2 \).  

We will now show that \texttt{Next-Fit} is worst possible fair algorithm. This will also show that the lower bound from Theorem 3.14 is tight.  

\textbf{Theorem 3.15.} For \( k \geq 4 \), the competitive ratio of \texttt{Next-Fit} on \( k \)-regular trees is \( \frac{k-2}{k-1} \).

\textit{Proof.} The adversary first gives a fan with \( k - 1 \) edges. It then gives \( N \) fans with \( k - 2 \) edges each. At the end, it gives again a fan with \( k - 1 \) edges. For \( i = 1, 2, \ldots, N \), the \( i \)'th fan is then connected to the \( i + 1 \)'th fan by a single edge. These connecting edges will all be rejected by \texttt{Next-Fit}. The graph is clearly \( k \)-regular and we get a competitive ratio of

\[
\frac{N(k - 2) + 2(k - 1)}{N(k - 2) + 2(k - 1) + N + 1},
\]

which tends to \( \frac{k-2}{k-1} \) as \( N \) tends to infinity.  

![Figure 3.8: The adversary graph for \( N = 2 \) used in Theorem 3.15. The dotted edges are rejected by \texttt{Next-Fit.}](image)

\textbf{Binary trees}

Since we have shown that the competitive ratio of any fair algorithm tends to one on \( k \)-colorable trees when \( k \) grows, it might be interesting to know what happens when \( k \) is small. In this section, we will look at the problem of coloring binary (or 3-colorable) trees with 3 colors available. This can be seen as continuing our previous analysis of paths (or 2-colorable trees).

We give a lower bound for fair algorithms. To this end, we use the method of counting from Theorem 2.6 but with an extension to account for the leafs in the tree.

\textbf{Lemma 3.16.} For any fair algorithm \( F \) with 3 colors available, the competitive ratio on 3-colorable trees is at least \( \frac{2}{3} \).

\textit{Proof.} Let \( T \) be a 3-colorable tree the edges of which is given to the fair algorithm \( F \) in some sequence. Denote by \( E_c \) the edges colored by \( F \) and by \( E_r \) the edges rejected by \( F \).
Figure 3.9: The three different cases for a rejected edge \((x, y)\) in a 3-colorable tree

As previous, each vertex \(x\) gets a value of \(\frac{1}{2}d_c(x)\). This value is then distributed equally among the rejected edges so that each rejected edge incident to \(x\) receives a value of \(\frac{1}{2} \cdot \frac{d_c(x)}{d_r(x)}\). Notice that even if \(x\) is a leaf it will always (since \(F\) is fair) receive a value of \(\frac{1}{2}\). Using Lemma 3.13 we try to prevent this value from going to waste by moving it to a rejected edge. Thus, if \(x\) is any vertex with \(d(x) = d_c(x) + d_r(x) > 2\) we get an extra value of \(\frac{1}{2} \cdot \frac{d_c(x) + d_r(x) - 2}{d_r(x)}\) from the associated leafs.

In total, the value \(v(x, y)\) given to a rejected edge \((x, y)\) is

\[
v(x, y) = \frac{1}{2} \frac{d_c(x)}{d_r(x)} + \frac{1}{2} \frac{d_c(x) + d_r(x) - 2}{d_r(x)} + \frac{1}{2} \frac{d_c(y)}{d_r(y)} + \frac{1}{2} \frac{d_c(y) + d_r(y) - 2}{d_r(y)}.
\]

Using the reasoning from Lemma 2.5, we see that if we can show \(v(x, y) \geq 2\) for all \((x, y) \in E_r\), then we get a lower bound on the competitive ratio of \(\frac{2}{3}\).

Consider a rejected edge \(e = (x, y)\). There are three possible cases to consider (plus the symmetric cases with \(x\) and \(y\) interchanged):

(i) \(d_c(x) = 2\) and \(d_c(y) = 2\). In this case, the edge \(e\) receives a value of

\[
v(x, y) = \frac{1}{2} \frac{2(2 + 1)}{1} + \frac{1}{2} \frac{2(2 + 1)}{1} = 3.
\]

(ii) \(d_c(x) = 2\), \(d_c(y) = 1\) and \(d_r(y) = 1\). In this case, the edge \(e\) receives a value of

\[
v(x, y) = \frac{1}{2} \frac{2(2 + 1)}{1} + \frac{1}{2} \frac{2}{1} = 2.
\]

(iii) \(d_c(x) = 2\), \(d_c(y) = 1\) and \(d_r(y) = 2\). In this case, the edge \(e\) receives a value of

\[
v(x, y) = \frac{1}{2} \frac{2(2 + 1)}{1} + \frac{1}{2} \frac{1(1 + 1)}{2} = 2.
\]

We conclude that \(v(x, y) \geq 2\) for all \(e = (x, y) \in E_r\). The desired lower bound on the competitive ratio of \(\frac{2}{3}\) follows from this. \(\blacksquare\)
A randomized algorithm on trees

We will now analyze the algorithm \texttt{Rand} on \( k \)-colorable trees.

**Lemma 3.17.** Consider the coloring produced by \texttt{Rand} on some graph \( G \). Let \( e \) be an edge in the graph which was not rejected by \texttt{Rand}. Then for all \( i = 1, 2, \ldots, k \), the probability that \( e \) received the color \( i \) is \( \frac{1}{k} \). Furthermore, the set of colors \( C_v \) used at a vertex \( v \) is selected uniformly random from subsets of \( C_{1,k} \) of size \( |C_v| \).

**Proof.** We prove the result by induction on the size of the graph. For \(|E| = 1\), the result is true by the definition of \texttt{Rand}. Assume now that the result holds for \(|E| = m\). Let \( G \) be a graph with \( m + 1 \) edges. For each of the first \( m \) edges given to \texttt{Rand}, we know it receives a uniformly random color if it is not rejected. Let \( e = (x, y) \) be the last (the \( m + 1 \)'th) edge given to \texttt{Rand}. If \( e \) is rejected there is nothing to prove, so suppose that \( e \) is colored.

Denote by \( C_x \) the colors used at \( x \) and by \( C_y \) the colors used at \( y \). By the induction hypothesis, both \( C_x \) and \( C_y \) are sets of colors selected uniformly at random from \( C_{1,k} \). Fix some \( 1 \leq i \leq k \). We want to calculate the probability that \( e \) gets the color \( i \). This is the probability that \( i \) is not used at \( x \) or \( y \) and that \texttt{Rand} selects \( i \) from the available colors:

\[
P(e \text{ gets color } i) = P(i \notin C_x \cup C_y \land \texttt{Rand} \text{ selects } i) = P(i \notin C_x \cup C_y) \cdot P(\texttt{Rand} \text{ selects } i | i \notin C_x \cup C_y)
= \frac{k - |C_x \cup C_y|}{k} \cdot \frac{1}{k - |C_x \cup C_y|}
= \frac{1}{k}.
\]

\( \Box \)

**Lemma 3.18.** Consider \( l, r, k \in \mathbb{N} \) satisfying \( l + r \geq k \) and \( l, r < k \). Let \( L, R \) be subsets of \( 1, \ldots, k \) of size \( l \) and \( r \), respectively. Assume that \( L \) and \( R \) are selected uniformly at random and independently from each other. Then the probability that \( L \cup R = 1, \ldots, k \) is

\[
\frac{ll!r!}{(l + r - k)!(k!)}.
\] (3.18.1)

**Proof.** Define \( K = 1, 2, \ldots, k \). Fix some \( l \)-subset \( L \) of \( K \). If \( L \cup R = K \), then the \( r \)-subset \( R \) must contain the \( k - l \) elements from \( K \) not in \( L \). The remaining \( r - (k - l) \geq 0 \) elements of \( R \) can be freely selected from \( L \). Thus, there are \( \binom{l}{r - (k - l)} \) subsets of \( K \) of size \( r \) such that \( L \cup R = K \). The probability we seek is therefore

\[
\frac{\binom{l}{r - (k - l)}}{\binom{k}{r}} = \frac{ll!r!}{(l + r - k)!(k)!}.
\]
Combining Lemma 3.17 and 3.18, we can calculate the probability that \textsc{Rand} rejects an edge \(e = (x, y)\) in a tree. Since a tree contains no cycles, the colors used at \(x\) and \(y\) are independent of each other up until the point where the edge \(e\) is given. It follows that if \(d_c(x) + d_c(y) \geq k\), then the probability of rejection is \(\frac{d_c(x)!d_c(y)!}{(d_c(x) + d_c(y) - k)!k!}\).

We are now ready to give an upper bound for the competitive ratio of \textsc{Rand} on \(k\)-regular trees.

**Lemma 3.19.** The competitive ratio of \textsc{Rand} is at most

\[
\frac{k^2 - 2}{k^2 + k - 4}
\]

on \(k\)-colorable trees for \(k \geq 3\).

We note that \(\frac{k+2}{k+3} \leq \frac{k^2-2}{k^2+k-4} \leq \frac{k+3}{k+4}\) for \(k \geq 4\).

**Proof.** The adversary graph is identical to the one used for \textsc{Next-Fit} in Theorem 3.15. The adversary first gives a fan with \(k-1\) edges. It then gives \(N\) fans with \(k-2\) edges each. At the end, it gives again a fan with \(k-1\) edges.

For \(i = 1, 2, \ldots, N\), the \(i\)’th fan is then connected to the \(i+1\)’th fan by a single edge. By Lemma 3.17, each color in \(C_{1,k}\) has probability \(\frac{k-2}{k}\) of appearing at a fan with \(k-2\) edges.

Consider the edges which may be rejected by \textsc{Rand}. For the analysis, the sequence in which the adversary gives these edges will be split into phases. A new phase begins whenever the adversary is going to connect a vertex with \(k-1\) colored edges incident to a fan with \(k-2\) colored edges. This is exactly the situation in the very beginning and also whenever we have just colored an edge connecting two fans. A phase ends when \textsc{Rand} colors one of the edges connecting two fans. Thus, in each complete phase there will be exactly one colored edge (the very last edge in the phase).

We will calculate the expected number of edges in a complete phase. For the first edge \(e = (x, y)\) in a phase, we have that \(d_c(x) = k-1\) and \(d_c(y) = k-2\) so that \(e\) has probability

\[
\frac{(k-1)!(k-2)!}{((k-1) + (k-2) - k)!k!} = \frac{k-2}{k},
\]

of being rejected. If it is colored, the phase ends with a length of one. If it is rejected, each of the following edges in the phase has probability

\[
\frac{(k-2)!(k-2)!}{((k-2) + (k-2) - k)!k!} = \frac{(k-3)(k-2)}{(k-1)k}
\]

of being rejected. For ease of notation, let \(p = \frac{(k-3)(k-2)}{(k-1)k}\). If the first edge is rejected, the number of additional edges needed before \textsc{Rand} is able to color
an edge is geometrically distributed. The expected value is \( \frac{1}{1-p} \). Thus, the expected length \( l \) of a complete phase is

\[
l = \frac{2}{k} \cdot 1 + \frac{k-2}{k} \cdot \left( \frac{1}{1-p} + 1 \right).
\]

This allows us to give an upper bound on the number of edges colored by Rand. It will color all the \( N \cdot (k - 2) + 2(k - 1) \) making up the fans. The number of connecting edges that Rand colors is exactly the number of complete phases. The expected number of complete phases is at most \( \frac{N}{k} \). We say at most since there might be an incomplete phase at the end. Thus, the competitive ratio of Rand on this graph is at most

\[
\frac{\frac{N}{k} + N(k - 2) + 2(k - 1)}{N(k - 2) + 2(k - 1) + N + 1}.
\]

If we insert the value of \( l \) and let \( N \) tend to infinity, the fraction tends to

\[
\frac{k^2 - 2}{k^2 + k - 4}.
\]
4 Conclusion and future work

In this report, the accommodating function has been determined for the algorithm Next-Fit for all values of $\alpha$ and for First-Fit when $\alpha \leq 1$. For $\alpha \leq 1$, the results are similar to the results obtained in [FN00] for $k$-colorable graphs. As already noted, there is a large gap between the best upper bound for deterministic algorithms and the value of the accommodating function for Next-Fit and First-Fit. This is true even when we consider only fair algorithms. I tried to close this gap by investigating the problem for small values of $k$. It turned out that First-Fit was actually optimal when $\alpha = 1$ and $k = 3$. But the adversary graph used is specifically designed to be 3-colorable and it is not clear if similar ideas can be used for larger values of $k$.

For $1 \leq \alpha$, the gap between the value of the accommodating function for Next-Fit and the upper bound of $\frac{1}{2}$ for fair algorithms is smaller. However, we still do not know if the competitive ratio of First-Fit is actually better than the competitive ratio of Next-Fit. It did not seem easier to me proving a lower bound on the accommodating function for First-Fit than proving a lower bound on the competitive ratio.

It was possible to completely solve the problem of edge coloring a path with respect to the competitive ratio. It seems particularly interesting that the fair, randomized algorithm $R_p$ achieves a better competitive ratio than any deterministic algorithm. Recall that for the minimum edge-coloring problem, randomization does not help (at least not on trees). I believe that randomization would help for the maximum edge-coloring problem in the general case. However, when the graph is allowed to have cycles it becomes difficult to calculate the probability of an edge being rejected. Different techniques than the ones from this work is needed to determine the competitive ratio of e.g. Rand.

It turned out that the max coloring problem is very easy on trees. In particular, any fair algorithm has a competitive ratio which tends to one as $k$ tends to infinity. On the other hand, all the adversary graphs we use (except for the one proving that First-Fit is optimal when $k = 3$ and $\alpha = 1$) are bipartite. Thus, it seems interesting to find other classes of graphs which are somewhere in between these two extremes.
Bibliography


