Abstract. In many applications in sequencing and scheduling it is desirable to have an underlying graph as equitably colored as possible. In this paper we survey recent theoretical results concerning conditions for equitable colorability of some graphs and recent theoretical results concerning the complexity of equitable coloring problem. Next, since the general coloring problem is strongly NP-hard, we report on practical experiments with some efficient polynomial-time algorithms for approximate equitable coloring of general graphs.

Keywords: computer experiments, corona graph, equitable chromatic number, equitable coloring conjectures, NP-hardness, polynomial heuristics
1. Introduction

All graphs considered in this paper are finite and simple, i.e. undirected, loopless and without multiple edges.

If the set of vertices of a graph $G$ can be partitioned into $k$ (possibly empty) classes $V_1, V_2, \ldots, V_k$ such that each $V_i$ is an independent set and the condition $||V_i| - |V_j|| \leq 1$ holds for every pair $(i, j)$, then $G$ is said to be equitably $k$-colorable. The smallest integer $k$ for which $G$ is equitably $k$-colorable is known as the equitable chromatic number of $G$ and is denoted by $\chi_{=}(G)$ [1]. Since equitable coloring is a proper coloring with an additional constraint, we have $\chi(G) \leq \chi_{=}(G)$ for any graph $G$.

This model of graph coloring has many practical applications. Every time we have to divide a system with binary conflict relations into equal or almost equal conflict-free subsystems we can model this situation by means of equitable graph coloring. One motivation for equitable coloring suggested by Meyer [1] concerns scheduling problems. In this application, the vertices of a graph represent a collection of tasks to be performed, and an edge connects two tasks that should not be performed at the same time. A coloring of this graph represents a partition of tasks into subsets that may be performed simultaneously. Due to load balancing considerations, it is desirable to perform equal or nearly-equal numbers of tasks in each time slot, and this balancing is exactly what an equitable coloring achieves. Furmańczyk [2] mentions a specific application of this type of scheduling problem, namely assigning university courses to time slots in a way that avoids scheduling incompatible pairs of courses at the same time and spreads the courses evenly among the available time slots, since then the usage of scarce additional resources (e.g. rooms) is maximalized.

The notion of equitable colorability was introduced by Meyer [1]. However, an earlier work of Hajnal and Szemerédi [3] showed that a graph $G$ with maximal degree $\Delta$ is equitably $k$-colorable if $k \geq \Delta + 1$. In the same paper [1] he formulated the following conjecture:

**Conjecture 1** (Equitable Coloring Conjecture (ECC)). For any connected graph $G$, other than complete graph or odd cycle, $\chi_{=}(G) \leq \Delta$.

This conjecture has been verified for all graphs on six or fewer vertices. Lih and Wu [4] proved that the Equitable Coloring Conjecture is true for all bipartite graphs. Wang and Zhang [5] considered a broader class of graphs, namely $r$-partite graphs. They proved that Meyer’s conjecture is true for complete graphs from this
class. Also, the conjecture was confirmed for outerplanar graphs [6] and planar graphs with maximum degree at least 13 [7].

There are very few papers on the complexity of equitable coloring. First of all, a straightforward reduction from graph coloring to equitable coloring by adding sufficiently many isolated vertices to a graph, proves that it is NP-complete to test whether a general graph has an equitable coloring with a given number of colors (greater than two). Secondly, Bodlaender and Fomin [8] showed that equitable coloring can be solved to optimality in polynomial time for trees (previously known due to Chen and Lih [9]) and outerplanar graphs. A polynomial time algorithm is also known for equitable coloring of split graphs [10].

This paper consists of two parts. In the first part (Section 2) we survey recent theorems and conjectures concerning sufficient conditions under which a graph is equitably \( r \)-colorable. In addition to this we survey recent theorems and conjectures concerning the complexity of optimal equitable coloring of certain simplified families of graphs. In the second part (Section 3) we report on practical experiments with some heuristics for optimal equitable coloring of general graphs. On the basis of these experiments we claim that the best results are obtained by using the SLF coloring algorithm with the FJK balancing procedure. We state a conjecture that this heuristic algorithm produces solutions which are almost surely not worse than \((2 + \epsilon)\chi_e(G)\) for any \(\epsilon > 0\).

2. Recent results and conjectures

In 1964 Erdös [11] conjectured that any graph \( G \) with maximum degree \( \Delta \leq k \) has an equitable \((k + 1)\)-coloring. This conjecture was proved in 1970 by Hajnal and Szemeredi [3] with a long and complicated proof. Kierstead et al. [12] found a polynomial-time algorithm of complexity \( O(\Delta^2 |V(G)|^2) \) for such a coloring. Recently, Kierstead and Kostochka [13] gave a short proof of this theorem, and presented another polynomial-time algorithm for such a coloring. Hajnal-Szemeredi’s bound has interesting applications in extremal combinatorial and probabilistic problems, see e.g. [14, 15].

Hajnal-Szemeredi’s bound, also named Dirac-type result was extended by Kierstead and Kostochka [16]. They proved the following Ore-type theorem.

**Theorem 1.** [16] *Every graph satisfying \( \text{deg}(x) + \text{deg}(y) \leq 2r + 1 \) for every edge \( \{x, y\} \), has an equitable \((r + 1)\)-coloring.*
Equitable coloring of graphs. Recent theoretical results.

Above theorem, like the Hajnal-Szemeredi theorem, is tight. Of course, these graphs for which the Hajnal-Szemeredi Theorem is tight simultaneously show that Theorem 1 is tight. However, there are more graphs for which this theorem is tight. For example, for every odd \( n < \Delta + 1 \), the graph \( K_{n,2\Delta+2-n} \) satisfies the inequality \( \deg(x) + \deg(y) \leq 2\Delta+2 \) for every edge \( \{x,y\} \) and has no equitable \((\Delta+1)\)-coloring. The authors of [16] conjectured the following hypothesis:

**Conjecture 2** ([16]). Let \( r \geq 3 \). If \( G \) is a graph satisfying \( \deg(x) + \deg(y) \leq 2r \) for every edge \( \{x,y\} \), and \( G \) has no equitable \( r \)-coloring, then \( G \) contains either \( K_{r+1} \) or \( K_{n,2r-n} \) for some odd \( n \).

The conjecture was proved for \( r = 3 \) [16]. It is natural to ask which graphs \( G \) with \( \Delta = r \geq 3 \) have equitable \( r \)-colorings. Certainly such graphs are \( r \)-colorable and do not contain \( K_{r+1} \). Since 1994 there has been known another basic conjecture formulated by Chen et al. [17]:

**Conjecture 3** (The Equitable \( \Delta \)-Coloring Conjecture (E\( \Delta \)CC)). A connected graph \( G \) is equitably \( \Delta \)-colorable if \( G \) is different from \( C_{2n+1} \), \( K_n \) and \( K_{2n+1,2n+1} \) for all \( n \geq 1 \).

It was proved for all graphs with \( \Delta \leq 4 \) [17, 18] as well as for some particular classes of graphs. Some special cases of Conjecture 3 were proved in [17, 4, 6, 7].

Chen and Yen [19] gave necessary conditions for a graph \( G \) (not necessarily connected) with \( \Delta \geq \chi(G) \) to be equitably \( \Delta \)-colorable and proved that those necessary conditions become sufficient when \( G \) is a bipartite graph, or \( G \) satisfies \( \Delta \geq \lfloor V(G)/3 \rfloor + 1 \), or \( G \) satisfies \( \Delta \leq 3 \). They supposed that the following is true.

**Conjecture 4** ([19]). Let \( G \) be a graph with \( \Delta \geq \chi(G) \). Then \( G \) is equitably \( \Delta \)-colorable if and only if at least one of the following statements holds.

1. \( \Delta \) is even.
2. No component or at least two components of \( G \) are isomorphic to \( K_{\Delta,\Delta} \).
3. Only one component of \( G \) is isomorphic to \( K_{\Delta,\Delta} \) and \( \alpha(G - K_{\Delta,\Delta}) > |V(G - K_{\Delta,\Delta})|/\Delta > 0 \).

Kierstead and Kostochka [20] introduced the notion of an \( r \)-equitable graph, as follows
Definition 2. Graph $G$ is $r$-equitable if

1. $|V(G)|$ is divisible by $r$
2. $G$ is $r$-colorable
3. Every $r$-coloring of $G$ is equitable.

For example, $K_r$ is $r$-equitable.

Conjecture 5 ([19]). Let graph $G$ satisfy $\Delta = r \geq \chi(G)$. Then $G$ has no equitable $r$-coloring if and only if $r$ is odd, $G$ has a subgraph $H$ isomorphic to $K_{r,r}$ and $G - H$ is $r$-equitable.

Chen et al. [21] proved that Conjecture 4 and Conjecture 5 are equivalent.

Definition 3. An $r$-equitable graph $G$ is $r$-reducible if $V(G)$ has a partition $\{V_1, \ldots, V_t\}$ into at least two parts such that the induced subgraph $G[V_i]$ is $r$-equitable for each $1 \leq i \leq t$; otherwise $G$ is $r$-irreducible.

Of course, $K_r$ is $r$-irreducible. Kierstead and Kostochka [20] identified ten other $r$-irreducible graphs ($r = 3, 4, 5$) (see Fig. 1). They named them $r$-basic graphs.

Definition 4. An $r$-decomposition of $G$ is a partition $\{V_1, \ldots, V_t\}$ of $V(G)$ such that each $G[V_i]$ is $r$-basic. The graph $G$ is $r$-decomposable if it has an $r$-decomposition.

It turns out that terms $r$-equitable and $r$-decomposable are equivalent for graphs $G$, where $\Delta = r$ and $r$ divides $|V(G)|$.

Conjecture 6 ([20]). Suppose that $\Delta = r \geq 3$ and $G$ is an $r$-colorable graph. Then $G$ is not equitably $r$-colorable if and only if the following conditions hold.

1. $r$ is odd.
2. $G$ has a subgraph $H = K_{r,r}$.
3. $G - H$ is $r$-decomposable.
Figure 1. $r$-basic graphs; $F_1: r = 5; F_2, F_3, F_4: r = 4; F_5, \ldots F_{10}: r = 3.$
It was proved [21, 20] that Kierstead-Kostochka’s Conjecture 6 is equivalent to EΔCC Conjecture and may help to prove it by induction.

Another research path in equitable coloring is determining exact values of equitable chromatic numbers and equitable thresholds for some specific graph classes, e.g. trees [9], complete multipartite graphs [22], Kneser graphs [23] and graph products. The last were intensively studied in [24, 2, 25] - Cartesian products, [2, 26, 27] - cross (Kronecker) products and in [28, 29, 30, 31] - corona products.

Among new results concerning corona products one deserves special attention. In [30] the problem for coronas of cubic graphs was considered. A cubical corona $G \circ H$ is a corona obtained by taking a cubic graph $G$ as the center graph and $|V(G)|$ copies of a cubic graph $H$ as the outer graph. The smallest cubical corona is shown in Fig. 2. Although the problem of ordinary coloring of coronas of cubic graphs is solvable in polynomial time, the problem of equitable coloring becomes intractable for these graphs. There have been given polynomially solvable cases of coronas of cubic graphs and the general problem of colorability of cubical coronas has been proved to be NP-hard. In this way the authors established a new class of graphs, for which equitable coloring problem is harder than that of ordinary coloring.

It is obvious that

$$2 \leq \chi_e(G) \leq 4,$$

for any cubic graph $G$. For each $k = 2, 3, 4$ let $\text{Cub}_k$ be the class of equitably $k$-chromatic cubic graphs. Let us notice that $\text{Cub}_4 = \{K_4\}$. Let $\text{Cub}_{3,u,v,w}^y \subset \text{Cub}_3$ be the class of 3-partite graphs with color classes of cardinalities $u$, $v$ and $w$, respectively. We have

**Theorem 5 ([30]).** The problem of deciding whether $\chi_e(K_{3,3} \circ H) = 4$ is NP-hard even if $H \in \text{Cub}_{3,u,v,w}^y$ and $y$ is divisible by 10.

The authors of [30] fixed all the cases of coronas of cubic graphs for which 3 colors suffice for equitable coloring. In the remaining cases they proved that 5 colors are enough for equitable coloring. These results are summarized in Table 1.

Moreover, a simple linear time algorithm for equitable coloring of such graphs which uses $\chi_e(G \circ H)$ or $\chi_e(G \circ H) + 1$ colors was obtained. This algorithm is best possible, unless $P = NP$. 
Equitable coloring of graphs. Recent theoretical results.

Figure 2. Corona $K_4 \circ K_4$.

Table 1. Possible values of the equitable chromatic number of coronas $G \circ H$.

<table>
<thead>
<tr>
<th>$G$</th>
<th>$H$</th>
<th>Cub$_2$</th>
<th>Cub$_3$</th>
<th>Cub$_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cub$_2$</td>
<td>3 or 4 $^1$</td>
<td>$4 \leq \chi_e \leq 5$ $^2$</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Cub$_3$</td>
<td>3 or 4 $^1$</td>
<td>$4 \leq \chi_e \leq 5$ $^2$</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Cub$_4$</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

$^1$: by this we mean that all cases when $\chi_e = 3$ or $\chi_e = 4$ are determined,
$^2$: deciding which value (4 or 5) is exact is NP-hard.

3. Computer experiments

In this section we consider two heuristics leading to equitable coloring of a given graph. Both of them are based on heuristics for classical coloring: Greedy coloring, LF, SL, SLF [32], and in both cases our algorithms transform classical coloring to equitable coloring. Our algorithms are based on two heuristics given in [24]. The first of them, called Naïve, relied on the principle of swapping colors of vertices colored with the most and the least frequently used colors. If this failed
a new color was introduced. This algorithm has been improved by checking all possible pairs of colors of vertices colored with the most and the least frequently used colors. New improved version, called FJK, outperforms its predecessor.

**Algorithm FJK(G);**

begin
  calculate frequency of colors;
  while coloring is not equitable do begin
    find vertices colored with
    the colors of maximal frequency;
    if it is possible to change the color of a vertex colored
    with the color of maximal frequency
to color of minimal frequency then do it
    else assign a new color to some vertex colored
    with color of maximal frequency;
  end
end;

To estimate the complexity of FJK we first show that the number of recolorings in this algorithm is $O(n \log n)$. Let us consider any color class $V_k$. After each step of the algorithm the cardinality of $V_k$ may increase by 1 or decrease by 1. The cardinality is increased if $V_k$ is the smallest color class. The cardinality is decreased if $V_k$ is the largest color class. Let sequence $(a^1_k, a^2_k, \ldots, a^s_k)$ denote the cardinalities of the color class $V_k$ during the course of the algorithm. It is easy to see that there exists $s_k$ such that for all $i < s_k$ $a^k_i \leq a^k_{i+1}$ and for all $i \geq s_k$ $a^k_i \geq a^k_{i+1}$. For each $i$ there cannot be more than $i$ color classes of size $\frac{n}{i}$. Therefore the maximal number of recolorings is less than

$$\sum_{k=2}^{n} \frac{n}{k} = O(n \log n)$$

It is easy to see that the estimation is very rough because we are not considering the graph structure neither the primary coloring. Since each recoloring takes time $O(n^2)$, so the complexity of the algorithm is $O(n^3 \log n)$.

The second heuristic, CreateSubgraph2, is based on CreateSubgraph from [24] and relies on the idea of swapping colors in entire subgraphs induced by vertices colored with the most and the least frequently used colors. The former
algorithm has been improved by checking all possible pairs of color classes with the most and the least frequently used colors. The algorithm swaps 2-colored subgraphs optimally, i.e. the difference between the number of vertices colored with the first color and the number of vertices with the second color is as small as possible. We skip a detailed description of CreateSubgraph2, since its results appeared worse than those of FJK.

Those two balancing procedures followed the four above-mentioned classical algorithms (In this way we obtained 8 particular algorithms). We tested them on random graphs of order $n = 100, 200, \ldots, 1000$ and densities $d = 0.1, 0.2, \ldots, 0.9$. For each fixed order $n$ and fixed density $d$ we generated 100 graphs at random. For each graph we tested each combination of the algorithms (72000 runs). The tests were performed on AMD Athlon 64 X2 5000+ with 2GB of RAM. Because of the size of the evidence we present herein only part of the results.

In Tables 2 and 3 we present average number of colors obtained by the algorithms. The main classical coloring is denoted by O, the FJK algorithm by A and the CreateSubgraph2 algorithm by B. As one can see, in most cases, the second algorithm does not give better result and, moreover, it runs in a longer time (cf. Table 4). Therefore, we consider only the FJK algorithm in the further part of this section. Moreover, the best results were obtained when the balancing procedure followed the SLF algorithm. Therefore, in Table 3 we report on computational results involving SLF.

### Table 2. Average numbers of colors used by: O - Greedy coloring, A - Greedy+ FJK algorithm and B - Greedy+CreateSubgraph2 algorithm.

<table>
<thead>
<tr>
<th>$n$</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>6.29</td>
<td>9.99</td>
<td>13.08</td>
<td>15.81</td>
<td>18.52</td>
<td>21.08</td>
<td>23.58</td>
<td>25.76</td>
<td>28.20</td>
<td>30.54</td>
</tr>
<tr>
<td>O</td>
<td>9.68</td>
<td>13.95</td>
<td>17.12</td>
<td>20.36</td>
<td>23.10</td>
<td>25.85</td>
<td>28.55</td>
<td>30.99</td>
<td>33.44</td>
<td>35.82</td>
</tr>
<tr>
<td>A</td>
<td>10.08</td>
<td>14.71</td>
<td>17.82</td>
<td>21.02</td>
<td>23.82</td>
<td>26.50</td>
<td>29.15</td>
<td>31.68</td>
<td>34.18</td>
<td>36.65</td>
</tr>
<tr>
<td>B</td>
<td>10.40</td>
<td>14.94</td>
<td>18.02</td>
<td>21.20</td>
<td>23.95</td>
<td>26.50</td>
<td>29.12</td>
<td>31.65</td>
<td>34.12</td>
<td>36.60</td>
</tr>
<tr>
<td>O</td>
<td>11.10</td>
<td>14.80</td>
<td>19.03</td>
<td>22.31</td>
<td>25.00</td>
<td>27.65</td>
<td>30.25</td>
<td>32.74</td>
<td>35.24</td>
<td>37.70</td>
</tr>
<tr>
<td>A</td>
<td>11.51</td>
<td>15.40</td>
<td>19.21</td>
<td>22.49</td>
<td>25.19</td>
<td>27.82</td>
<td>30.40</td>
<td>32.89</td>
<td>35.39</td>
<td>37.85</td>
</tr>
<tr>
<td>B</td>
<td>11.92</td>
<td>15.82</td>
<td>19.39</td>
<td>22.57</td>
<td>25.28</td>
<td>27.92</td>
<td>30.50</td>
<td>32.99</td>
<td>35.49</td>
<td>37.95</td>
</tr>
<tr>
<td>O</td>
<td>12.55</td>
<td>16.00</td>
<td>19.57</td>
<td>22.76</td>
<td>25.47</td>
<td>28.09</td>
<td>30.67</td>
<td>33.16</td>
<td>35.66</td>
<td>38.13</td>
</tr>
<tr>
<td>A</td>
<td>13.00</td>
<td>16.70</td>
<td>20.01</td>
<td>23.01</td>
<td>25.71</td>
<td>28.32</td>
<td>30.90</td>
<td>33.39</td>
<td>35.89</td>
<td>38.36</td>
</tr>
<tr>
<td>B</td>
<td>13.45</td>
<td>17.01</td>
<td>20.22</td>
<td>23.26</td>
<td>25.96</td>
<td>28.52</td>
<td>31.05</td>
<td>33.54</td>
<td>36.04</td>
<td>38.51</td>
</tr>
<tr>
<td>O</td>
<td>14.55</td>
<td>17.00</td>
<td>20.29</td>
<td>23.37</td>
<td>26.05</td>
<td>28.55</td>
<td>31.05</td>
<td>33.55</td>
<td>36.05</td>
<td>38.55</td>
</tr>
<tr>
<td>A</td>
<td>15.05</td>
<td>17.50</td>
<td>20.39</td>
<td>23.52</td>
<td>26.15</td>
<td>28.65</td>
<td>31.15</td>
<td>33.65</td>
<td>36.15</td>
<td>38.65</td>
</tr>
<tr>
<td>B</td>
<td>15.55</td>
<td>17.90</td>
<td>20.50</td>
<td>23.67</td>
<td>26.25</td>
<td>28.75</td>
<td>31.25</td>
<td>33.75</td>
<td>36.25</td>
<td>38.75</td>
</tr>
</tbody>
</table>

For each fixed order $n$ and fixed density $d$ we generated 100 graphs at random. For each graph we tested each combination of the algorithms (72000 runs). The tests were performed on AMD Athlon 64 X2 5000+ with 2GB of RAM. Because of the size of the evidence we present herein only part of the results.

In Tables 2 and 3 we present average number of colors obtained by the algorithms. The main classical coloring is denoted by O, the FJK algorithm by A and the CreateSubgraph2 algorithm by B. As one can see, in most cases, the second algorithm does not give better result and, moreover, it runs in a longer time (cf. Table 4). Therefore, we consider only the FJK algorithm in the further part of this section. Moreover, the best results were obtained when the balancing procedure followed the SLF algorithm. Therefore, in Table 3 we report on computational results involving SLF.
Table 3. Average numbers of colors used by: O - SLF algorithm, A - SLF+FJK algorithm and B - SLF+CreateSubgraph2 algorithm.

<table>
<thead>
<tr>
<th>d</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>408</td>
<td>413</td>
<td>418</td>
<td>424</td>
<td>430</td>
<td>436</td>
<td>442</td>
<td>448</td>
<td>454</td>
<td>460</td>
</tr>
<tr>
<td>0.2</td>
<td>515</td>
<td>525</td>
<td>535</td>
<td>545</td>
<td>555</td>
<td>565</td>
<td>575</td>
<td>585</td>
<td>595</td>
<td>605</td>
</tr>
<tr>
<td>0.3</td>
<td>622</td>
<td>632</td>
<td>642</td>
<td>652</td>
<td>662</td>
<td>672</td>
<td>682</td>
<td>692</td>
<td>702</td>
<td>712</td>
</tr>
<tr>
<td>0.4</td>
<td>729</td>
<td>740</td>
<td>750</td>
<td>760</td>
<td>770</td>
<td>780</td>
<td>790</td>
<td>800</td>
<td>810</td>
<td>820</td>
</tr>
<tr>
<td>0.5</td>
<td>837</td>
<td>850</td>
<td>860</td>
<td>870</td>
<td>880</td>
<td>890</td>
<td>900</td>
<td>910</td>
<td>920</td>
<td>930</td>
</tr>
<tr>
<td>0.6</td>
<td>948</td>
<td>960</td>
<td>970</td>
<td>980</td>
<td>990</td>
<td>1000</td>
<td>1010</td>
<td>1020</td>
<td>1030</td>
<td>1040</td>
</tr>
<tr>
<td>0.7</td>
<td>1051</td>
<td>1064</td>
<td>1074</td>
<td>1084</td>
<td>1094</td>
<td>1104</td>
<td>1114</td>
<td>1124</td>
<td>1134</td>
<td>1144</td>
</tr>
<tr>
<td>0.8</td>
<td>1157</td>
<td>1170</td>
<td>1180</td>
<td>1190</td>
<td>1200</td>
<td>1210</td>
<td>1220</td>
<td>1230</td>
<td>1240</td>
<td>1250</td>
</tr>
<tr>
<td>0.9</td>
<td>1274</td>
<td>1287</td>
<td>1300</td>
<td>1310</td>
<td>1320</td>
<td>1330</td>
<td>1340</td>
<td>1350</td>
<td>1360</td>
<td>1370</td>
</tr>
</tbody>
</table>

Table 4. Average computation time of algorithms [ms].

<table>
<thead>
<tr>
<th>d</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
<th>700</th>
<th>800</th>
<th>900</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td>65</td>
</tr>
<tr>
<td>0.2</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
</tr>
<tr>
<td>0.3</td>
<td>40</td>
<td>45</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>80</td>
<td>85</td>
</tr>
<tr>
<td>0.4</td>
<td>50</td>
<td>55</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>80</td>
<td>85</td>
<td>90</td>
<td>95</td>
</tr>
<tr>
<td>0.5</td>
<td>60</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>80</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>105</td>
</tr>
<tr>
<td>0.6</td>
<td>70</td>
<td>75</td>
<td>80</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>105</td>
<td>110</td>
<td>115</td>
</tr>
<tr>
<td>0.7</td>
<td>80</td>
<td>85</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>105</td>
<td>110</td>
<td>115</td>
<td>120</td>
<td>125</td>
</tr>
<tr>
<td>0.8</td>
<td>90</td>
<td>95</td>
<td>100</td>
<td>105</td>
<td>110</td>
<td>115</td>
<td>120</td>
<td>125</td>
<td>130</td>
<td>135</td>
</tr>
<tr>
<td>0.9</td>
<td>100</td>
<td>105</td>
<td>110</td>
<td>115</td>
<td>120</td>
<td>125</td>
<td>130</td>
<td>135</td>
<td>140</td>
<td>145</td>
</tr>
</tbody>
</table>

Also, we compared results obtained by Greedy (classical coloring) and SLF+FJK (equitable coloring) algorithms. The results are summarized in Table 5 and shown in Figure 3. In 88% of cases the number of colors used by the latter is not greater than those by Greedy coloring. In the remaining 12% cases the number of colors used by SLF+FJK exceeds that of Greedy coloring by at most 20%, and the average is 3.8%.

It is known that Greedy algorithm is practically 2-approximate. More precisely, Grimmet and McDiarmid [33] showed that the Greedy algorithm when applied to a random ordering of the vertices yields a Greedy(G)-coloring for any
Figure 3. Greedy vs SLF+FJK algorithms - average numbers of colors used.

Table 5. Greedy vs SLF+FJK algorithm. The notion $X/Y/Z$ denotes that in $X$ cases Greedy algorithm used fewer colors, in $Y$ cases the both algorithms used the same number of colors and in $Z$ cases the algorithm SLF+FJK was better.

<table>
<thead>
<tr>
<th>n</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.891</td>
<td>6/14/89</td>
<td>4/8/88</td>
<td>11/10/79</td>
<td>7/15/78</td>
<td>14/7/79</td>
<td>1/5/39</td>
<td>12/11/77</td>
<td>1/1/98</td>
</tr>
<tr>
<td>200</td>
<td>0.397</td>
<td>8/5/89</td>
<td>5/8/87</td>
<td>5/4/91</td>
<td>7/20/73</td>
<td>5/18/77</td>
<td>12/14/74</td>
<td>9/0/39</td>
<td>0/0/100</td>
</tr>
<tr>
<td>300</td>
<td>0.294</td>
<td>5/11/84</td>
<td>3/4/93</td>
<td>7/12/81</td>
<td>2/2/96</td>
<td>14/4/82</td>
<td>1/3/80</td>
<td>2/1/99</td>
<td>0/0/100</td>
</tr>
<tr>
<td>500</td>
<td>0/096</td>
<td>3/1/82</td>
<td>1/6/93</td>
<td>4/10/86</td>
<td>21/8/71</td>
<td>3/5/92</td>
<td>2/8/98</td>
<td>0/3/97</td>
<td>2/2/96</td>
</tr>
<tr>
<td>600</td>
<td>0/091</td>
<td>4/5/91</td>
<td>1/7/92</td>
<td>9/1/90</td>
<td>9/0/39</td>
<td>13/21/66</td>
<td>1/4/95</td>
<td>2/7/91</td>
<td>24/10/66</td>
</tr>
<tr>
<td>700</td>
<td>0/095</td>
<td>0/5/91</td>
<td>0/1/100</td>
<td>8/3/89</td>
<td>1/1/98</td>
<td>48/8/44</td>
<td>22/27/51</td>
<td>37/18/50</td>
<td>14/6/18</td>
</tr>
<tr>
<td>800</td>
<td>0/089</td>
<td>0/4/96</td>
<td>0/11/89</td>
<td>3/6/91</td>
<td>6/4/80</td>
<td>31/0/66</td>
<td>69/10/21</td>
<td>62/6/27</td>
<td>68/0/32</td>
</tr>
<tr>
<td>900</td>
<td>0/091</td>
<td>0/3/97</td>
<td>3/9/88</td>
<td>10/17/73</td>
<td>4/17/79</td>
<td>15/1/84</td>
<td>56/0/44</td>
<td>73/0/27</td>
<td>34/0/66</td>
</tr>
<tr>
<td>1000</td>
<td>0/096</td>
<td>0/2/93</td>
<td>5/2/93</td>
<td>9/0/91</td>
<td>35/14/51</td>
<td>5/1/94</td>
<td>46/0/54</td>
<td>51/0/49</td>
<td>19/0/81</td>
</tr>
</tbody>
</table>

$\epsilon > 0$, where

$$\text{Greedy}(G) \leq (2 + \epsilon)\chi(G)$$

for all but a vanishingly small fraction of graphs as the number of vertices tends to infinity.

We conjecture that our FJK algorithm yields the same guarantee.

**Conjecture 7.**

$$\text{SLF+FJK}(G) \leq (2 + \epsilon)\chi(G)$$

for almost every graph $G$.

All the algorithms used in tests can be found in the KOALA library (kaims.pl/koala).
References


