Complexity of coloring random graphs: an experimental study of the hardest region*

Zoltán Ádám Mann

Abstract

It is known that the problem of deciding k-colorability of a graph exhibits an easy-hard-easy pattern, i.e., the average-case complexity for backtrack-type algorithms, as a function of k, has a peak. This complexity peak is either at $k = \chi - 1$ or $k = \chi$, where χ is the chromatic number of the graph. However, the behavior around the complexity peak is poorly understood. In this paper, we use list coloring to model coloring with a fractional number of colors between $\chi - 1$ and χ. We present a comprehensive computational study on the complexity of backtrack-type graph coloring algorithms in this critical range. According to our findings, an easy-hard-easy pattern can be observed on a finer scale between $\chi - 1$ and χ as well. The highest complexity found this way can be higher than for any integer value of k. It turns out that the complexity follows an alternating 3-dimensional pattern; understanding this pattern is very important for benchmarking purposes. Our results also answer the previously open question whether coloring with $\chi - 1$ or with χ colors is harder: this depends on the location of the maximal fractional complexity.

1 Introduction

Graph coloring is an important combinatorial optimization problem with many applications in engineering, such as register allocation, frequency assignment, pattern matching and scheduling [12, 40, 33, 38, 6]. Accordingly, graph coloring has been the subject of intensive research.

Deciding k-colorability of a graph is an NP-complete problem. Nevertheless, there are exact algorithms – mostly based on backtrack search – that can often solve even quite big problem instances in reasonable time. In sharp contrast to their exponential worst-case complexity, the average-case complexity of such algorithms can be polynomial or even constant [52, 35]. Such results are established by assuming a probability distribution on the set of possible inputs and calculating the expected number of steps of the algorithm.

If the average-case complexity of an algorithm is much lower than its worst-case complexity, this suggests that there are huge differences in the algorithm's running time on problem instances of the same size. This is certainly true for backtrack-style graph coloring algorithms. Since this variability of the runtime significantly hinders the practical adoption of such algorithms [37], considerable effort has been invested into understanding the parameters that influence the runtime, both theoretically and empirically (see the next section for details). As a result, it is known that algorithm runtime is mostly determined by the density of the graph and the number of available colors.

The dependence on the graph's density is quite well understood. A typical example is shown in Figure 1. Here, the number of vertices n and the number of available colors k are fixed, and each pair of vertices is connected by an edge with probability p. For small values of p, the graphs usually contain few edges, so that the fraction of k-colorable graphs is almost 1, and backtrack search can very quickly find a valid coloring. For high values of p, the graphs are mostly very dense, so that the fraction of k-colorable graphs is almost 0, and backtrack search can quickly establish non-k-colorability. There is a narrow range of p values in between, where a so-called phase transition occurs: the ratio of k-colorable instances changes abruptly from almost 1 to almost 0, accompanied by a striking peak of the algorithm's runtime. As n grows, the

*Published in *ACM Journal of Experimental Algorithmics*, 23(1): article 1.3, 2018

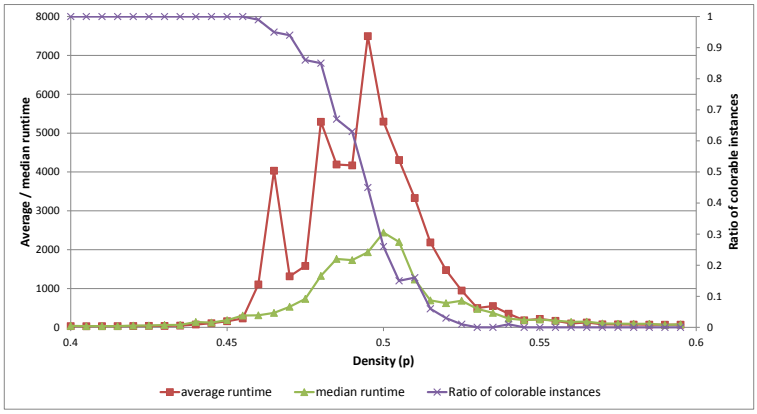


Figure 1: Colorability and average-case complexity as a function of the graph's density. This plot shows the average results on 100 random graphs with $n = 60$ vertices and $k = 10$ colors.

peak in algorithm runtime becomes higher and sharper. Similar phenomena have also been observed in the context of other NP-complete problems, e.g., Boolean satisfiability [27, 1], where the ratio of the number of clauses to the number of variables plays an analogous role to p.

It is also interesting to recognize in Figure 1 that in the critical region the average runtime is considerably higher than the median runtime. This can be attributed to the heavy-tailed runtime distribution that is typical of exact algorithms for NP-hard problems: utterly long runs occur with non-negligible probability, significantly increasing the average runtime but only marginally the median [24]. Moreover, in the left half of the critical region, the average runtime oscillates wildly. A possible explanation is that in the left half of the critical region, almost all problem instances are solvable, and in such cases fortunate decisions can lead the search very quickly to a solution, whereas other, less fortunate runs can take much more time. In contrast, for unsolvable instances, all branches of the search tree must be investigated, thus limiting the impact of chance. This is why no similar oscillation can be observed in the right half of the critical region.

The dependence on the number of colors k is a bit different. Clearly, a graph G is k-colorable if $k \geq \chi(G)$ and non-k-colorable if $k \leq \chi(G) - 1$, where $\chi(G)$ is the chromatic number of the graph. Thus, the transition between non-solvable and solvable instances is between $k = \chi - 1$ and $k = \chi$. One expects that the complexity[1] also peaks at either $k = \chi - 1$ or $k = \chi$. Some authors stated that the complexity peaks at $k = \chi - 1$ [26]. However, we showed in [49] that this is not always the case: peaks at $k = \chi - 1$ and $k = \chi$ both occur. In that work we also showed that for $k \leq \chi - 1$, complexity is monotonously increasing in k for a wide family of backtrack search algorithms; for $k \geq \chi$, complexity is usually, but not always, decreasing in k. (The latter phenomenon corresponds to the oscillations in the left part of the critical region in Figure 1.) However, whether complexity peaks at $\chi - 1$ or χ seems to be unpredictable.

An example is shown in Figure 2. At first sight, the diagram is very similar to Figure 1: both of them exhibit a clear complexity peak in the critical phase transition region, also characterized by a big gap between average and median runtime. Of course, the two diagrams are mirrored in the sense that colorable instances are in the left part of Figure 1 and in the right part of Figure 2, whereas uncolorable instances are in the right part of Figure 1 and in the left part of Figure 2.

However, there are also some peculiarities of Figure 2 that become especially interesting when compared with Figure 3, showing analogous diagrams for slightly perturbed parameter configurations.

- In contrast to Figure 1, no oscillations of algorithm runtime can be observed in the colorable case in Figures 2 and 3. Actually, since the complexity is already almost 0 for $k = 13$, our observations of the colorable case in the critical region are practically limited to $k = 11$ and $k = 12$, which makes it impossible to establish whether or not there are oscillations in this range.

[1] In this paper, unless otherwise specified, 'complexity' refers to the empiric average-case complexity for backtrack-type algorithms.

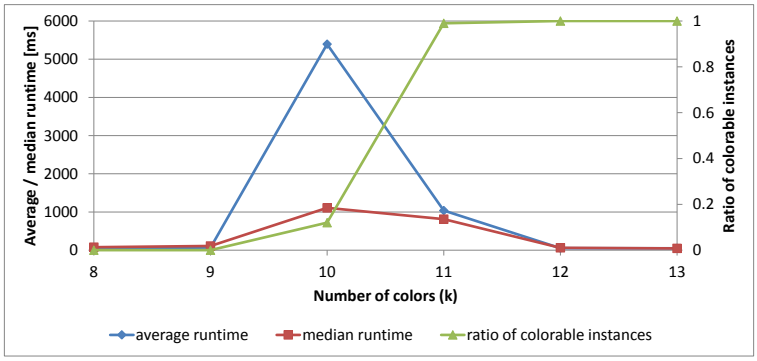


Figure 2: Colorability and average-case complexity as a function of the number of colors. This plot shows the average results on 100 random graphs with $n = 60$ vertices and edge probability $p = 0.51$.

- In all diagrams it is clear that most of the investigated graphs have a chromatic number of 11. On the other hand, the complexity peak is at 10 in Figures 2 and 3(b); in Figure 3(a) however, it is at 11.
- Although the graphs considered in Figure 3(b) have a higher number of vertices and higher expected number of edges than the ones of Figure 2, the complexity peak – both in terms of average and median runtime – is smaller in Figure 3(b) than in Figure 2 (note the different scale on the left vertical axis in the two figures).

The aim of this paper is to analyze the dependence of the experimental average-case complexity for backtrack-type algorithms on the number of colors in more depth, so that the above peculiarities can be better understood. The starting point is the following hypothesis: *the dependence on k is very similar to the dependence on p, but it is distorted by the fact that we can only measure it for integer values of k.*

Therefore, we extend the notion of k-colorability to fractional values of k. This is done by means of list coloring, inspired by the $(2 + p)$-coloring problem that was used by Walsh to explore the boundary between polynomially solvable and NP-hard problems [51]. With this machinery, we can draw fine-grained diagrams of the dependence on k, allowing us to support our hypothesis and the following findings:

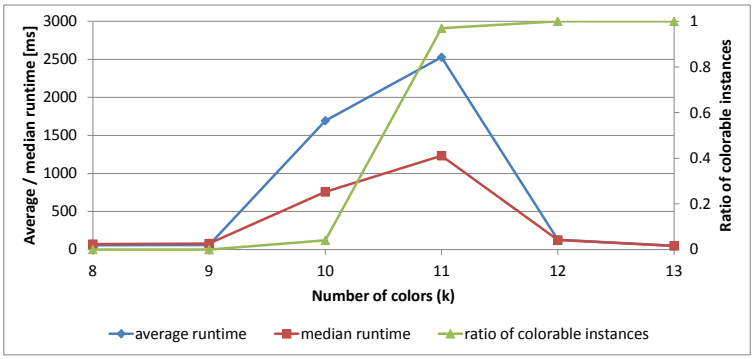
- The dependence on the fractional number of colors is much more predictable than the discretized version with values only for integer k. In particular, small changes in the parameters lead to small changes in the curves.
- The complexity peak is in the $[\chi - 1, \chi]$ interval. The runtime at this fractional k can be higher than the runtime in $\chi - 1$ or in χ.
- To really understand how the complexity peak moves with changing parameters, one must investigate it in a multi-dimensional way. This reveals an alternating sequence of peaks and valleys.

This better understanding of the problem's complexity can be directly used for the purposes of benchmarking: generating problem instances with a desired level of complexity [46]. Moreover, in the long run it may pave the way for devising algorithms with improved typical-case complexity.

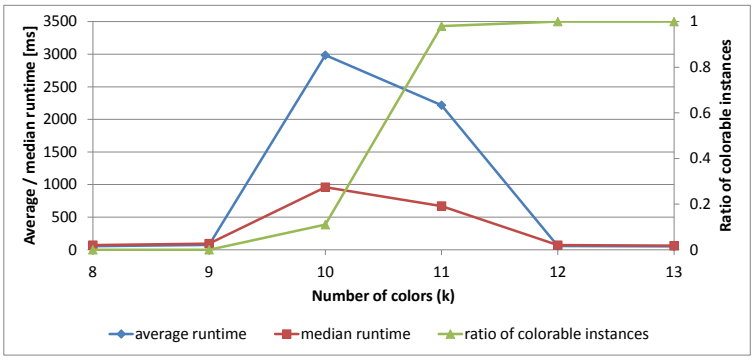
The rest of the paper is organized as follows. Section 2 discusses related work. After defining the basic notions in Section 3, Section 4 describes our methodology. Section 5 presents our empirical findings, followed by a discussion of potential threats to the validity of our findings in Section 6, while Section 7 concludes the paper.

2 Previous work

Because of its importance, the study of the complexity of graph coloring started already in the early 1970s. In fact, graph coloring was one of the 21 combinatorial problems whose NP-completeness was shown by



(a) $n = 60, p = 0.52$



(b) $n = 61, p = 0.51$

Figure 3: Colorability and average-case complexity as a function of the number of colors: small perturbations in the parameters.

Karp in his seminal 1972 paper [30]. Afterwards, researchers' attention turned towards approximation algorithms, but it turned out quickly that approximating the chromatic number is a hard problem. An early result of Garey and Johnson showed that no polynomial-time approximation algorithm with an approximation ratio smaller than 2 can exist, unless P=NP [23]. More recently, it was shown that – under standard assumptions of complexity theory – not even an $O(n^{1-\varepsilon})$ approximation can exist for any $\varepsilon > 0$ [22, 54].

Also starting with the 1970s, different heuristic and exact algorithms were developed for the graph coloring problem (see e.g. [13, 39, 11]). The proposed exact algorithms mostly used some form of backtrack search to guarantee a complete search while also being able to prune potentially large parts of the search space.

With the availability of practical graph coloring algorithms implemented as computer programs, researchers started to gain empirical experience with graph coloring in practice [11, 16, 18, 49]. This resulted in the discovery of the already mentioned phase transition phenomenon with the accompanying easy-hard-easy pattern [16, 28, 27, 18].

The more recent results mostly fall into one of two categories: (i) relating to the locus of the phase transition (i.e., the chromatic number), and (ii) relating to algorithm complexity. In both categories, there are both rigorously proven and empirically established results.

2.1 Results on the chromatic number

The analysis of the chromatic number of random graphs was first suggested in the seminal 1960 paper of Erdős and Rényi [21]. Subsequent work of Grimmett and McDiarmid [25], Bollobás [9], and Luczak [31], lead to an understanding of the order of magnitude of the expected chromatic number of random graphs: it is $\Theta(n/\log_b(n))$, where $b = 1/(1-p)$. Through the work of Shamir and Spencer [47], Luczak [32],

Alon and Krivelevich [5], and Achlioptas and Naor [4], we can determine almost exactly the expected chromatic number of a sparse random graph in the limit. For dense random graphs, much less is known: although the result of Shamir and Spencer guarantees that the chromatic number is concentrated in an interval of length $O(\log n)$ [47], the difference between the best known lower and upper bounds is still $\Theta(n(\log n)^{-2})$ [10, 45].

Upper bounds on the chromatic number were often proven in an algorithmic way, by showing that a specific (usually quite simple) algorithm will succeed in coloring the graph with high probability. An example is the GIS (Greedy Independent Set) heuristic that works by determining independent sets greedily and using them as color classes [25, 48, 19]. Other examples are the greedy list-coloring algorithm k-GL that selects a vertex with minimum number of available colors to be colored next [2], and its refinement in which ties are broken in such a way that vertices with more uncolored neighbors are selected with higher probability [3]. A possible interpretation of these results is that, for small constraint densities, the solution can be found without backtracking with positive probability [29]. In a similar way, Turner proved that the No-Choice algorithm – which, after coloring a clique, colors only vertices whose color is uniquely determined – finds a coloring for almost all dense k-colorable graphs, if $k = O(\log n)$ [50].

2.2 Results on algorithm complexity

Systematic experimental studies revealed in the 1990s that problem instances at the phase transition boundary tend to be considerably more difficult than elsewhere [16, 28, 27].

The asymptotics of the expected number of steps taken by exact graph coloring algorithms was also investigated rigorously [52, 8, 36]. These works either focused on the non-k-colorable case or assumed that the algorithm must find all solutions in the k-colorable case, since algorithms that terminate at the first found solution are much harder to analyze mathematically because of the dependence between random variables [35].

Interestingly, methods from theoretical physics (more specifically, statistical mechanics) have also been applied successfully to study the asymptotic expected performance of backtrack algorithms. After first results on the satisfiability problem [17], this machinery was also used to study the 3-coloring problem. In particular, Monasson and co-workers modeled the solution process of backtrack search with an out-of-equilibrium (multi-dimensional) surface growth problem [20, 42]. By solving the resulting partial differential equation, an estimation of the backtrack algorithm's runtime can be obtained that is fairly close to the empirical results for relatively dense graphs. Although these results are not rigorous, Monasson later developed a method based on generating functions, with which similar results were achieved in a rigorous way [43].

The phenomena relating to the complexity of graph coloring are shared by many other constraint satisfaction problems as well [34]. There have been promising attempts to explain the hardness of constraint satisfaction problem instances in certain parameter ranges. An example is the notion of backdoors: small sets of variables so that setting them in the right way makes the rest of the problem easy [53]. Another example are backbones: sets of variables that must have given values in all solutions [44]. In the latter case, one speaks about frozen variables; the fraction of frozen variables abruptly jumps from $o(n)$ to $\Theta(n)$ at the so-called freezing threshold, which is a density below the phase transition [41]. Asymptotically at the same density is the so-called clustering threshold, where the solution space shatters from one big connected cluster to an exponential number of small clusters [1]. These freezing and clustering phenomena seem to make it much harder to find a valid coloring of the graph.

Most previous work analyzed these phenomena as a function of the graph's density. In contrast, we investigate the dependence on the number of colors, and analyze it by means of a continuous relaxation. To our knowledge, this has not been done before.

3 Preliminaries

We consider the decision version of the graph coloring problem (denoted as k-colorability or k-COL), in which the input consists of an undirected graph $G = (V, E)$ and a number $k \in \mathbb{Z}^+$, and the task is to

decide whether the vertices of G can be colored with k colors such that adjacent vertices are not assigned the same color.

The input graph is a random graph, for which three models will be considered. In the $G_{n,p}$ model, the graph has n vertices and each pair of vertices is connected by an edge with probability p independently from each other. In the $G_{n,m}$ model, the graph has n vertices and m edges selected randomly, according to a uniform distribution (parallel edges and loops are not allowed). In the latter case, the density of the graph is defined as

$$p = \frac{m}{\binom{n}{2}}. \tag{1}$$

Obviously, if we take a graph from $G_{n,p}$ with this p value, then the expected value of the number of edges is exactly m. Hence, formula (1) correctly establishes the connection between the two models.

The third graph model, called R-MAT, has been suggested by Chakrabarti et al. to model real-world networks with power-law degree distribution, community structure, and small diameter [15]. The model is parameterized with four parameters $0 \leq a, b, c, d \leq 1$, where $a + b + c + d = 1$. The adjacency matrix of the graph is created from an empty graph iteratively with the following procedure. The adjacency matrix is split into four disjoint square submatrices of equal size. To create a random edge, first one of the sub-matrices is selected, where the probability of selecting each sub-matrix is a, b, c, d, respectively. Then the same procedure is repeated for the chosen sub-matrix, which is again split into four sub-matrices etc. The recursion ends when a single cell of the matrix is reached. If the cell contains 0 and is not in the main diagonal of the adjacency matrix, then it is switched to 1. The given number of edges is created this way.

The vertices of the graph will be denoted by $v_1, \ldots, v_n$, the colors by $1, \ldots, k$. A *coloring* c assigns a color to each vertex; the color of vertex v is $c(v)$.

The *chromatic number* of a graph G, denoted by $\chi(G)$, is the smallest k such that G is colorable with k colors.

We will also deal with the *list coloring problem*, in which the input consists of an undirected graph $G = (V, E)$, a number $k \in \mathbb{Z}^+$, and a list of lists L_i, $i = 1, 2, \ldots, n$, where each list L_i is a subset of $\{1, 2, \ldots, k\}$. The question is whether G admits a coloring c with k colors such that adjacent vertices are not assigned the same color and for each $v_i \in V$, $c(v_i) \in L_i$.

4 Methodology

Our aim is to assess the complexity of graph coloring using a fractional number of colors. For this, we need to (i) define what coloring with a fractional number of colors means and (ii) present algorithms to solve this problem. These two issues are addressed in the following two subsections, respectively.

4.1 Fractional coloring

Our approach is inspired by the work of Walsh on the $(2 + p)$-COL problem [51]. In $(2 + p)$-COL, where $0 < p < 1$, there are 3 colors; a fraction $1 - p$ of the vertices may be colored with only the first 2 colors, and a fraction p may be colored with any of the 3 available colors. Walsh used this technique to interpolate between the polynomially solvable 2-coloring problem and the NP-complete 3-coloring problem.

Following the same logic, we define the k-COL problem for $k \in \mathbb{R}, k \geq 1$ as follows. If $k \in \mathbb{Z}$, then the definition remains the same as in the normal k-colorability problem for integer k. Otherwise, let $k_1 = \lfloor k \rfloor$ and $k_2 = \lceil k \rceil = k_1 + 1$; furthermore, let $n_1 = [n \cdot (k_2 - k)]$ and $n_2 = n - n_1$. Here, $\lfloor x \rfloor$ is the greatest integer below x, $\lceil x \rceil$ is the least integer above x, and $[x]$ is the nearest integer according to the standard rules of rounding. Then, by coloring with k colors, we mean the list coloring problem with k_2 colors, in which n_1 vertices have the first k_1 colors available and the remaining n_2 vertices have all the k_2 colors available.

The n_1 vertices with k_1 colors can be chosen randomly. Alternatively, in the $G_{n,p}$ and $G_{n,m}$ models, we can simply take the first n_1 vertices. Hence here we can assume without loss of generality that

$$L_i = \begin{cases} \{1, 2, \ldots, k_1\} & \text{if } i \leq n_1 \\ \{1, 2, \ldots, k_2\} & \text{if } i > n_1. \end{cases}$$

This way, when k approaches k_1, the majority of vertices will have only the first k_1 colors available, so that the problem naturally approaches the normal coloring problem with $k_1 \in \mathbb{Z}^+$ colors (and similarly, when k approaches k_2, the problem approaches the normal coloring problem with $k_2 \in \mathbb{Z}^+$ colors). It is worth mentioning that we could define the coloring problem with a fractional number of colors in such a way that the first n_1 vertices have *some* k_1 available colors, randomly chosen for each of these vertices. However, this would not be a good idea because even if $k = k_1$ and thus $n_1 = n$, the resulting problem would not be equivalent to the standard coloring problem with an integer number of colors. As an example, consider a cycle of length 3. If each vertex has only the colors $L_1 = L_2 = L_3 = \{1, 2\}$ available, the graph is obviously not colorable. However, if $L_1 = L_2 = \{1, 2\}$ but $L_3 = \{2, 3\}$, then the graph is colorable. Hence it is important that the set of available colors for the first n_1 vertices consists of the *first* k_1 colors.

Another issue that should be noted is that, since we defined n_1 as $[n \cdot (k_2 - k)]$, this means that k should be increased in steps of $1/n$ in order to increase n_1 one by one. In other words, the precision that we can achieve when investigating the dependence on k is $\Delta k = 1/n$. For example, for $n = 50$, we can plot the dependence on k with steps of 0.02. Fortunately, this precision is sufficient for our purposes.

4.2 List coloring algorithms

In order to decrease the dependence of our results on a specific algorithm, we used three different list coloring algorithms.

4.2.1 Direct coloring algorithm

The first one is similar to the coloring algorithm presented in [49]. It is a backtrack search algorithm. It traverses the space of partial color assignments in a tree-like manner. For each vertex that has no color yet, the set of possible colors is maintained, which is initially the corresponding L_i list. In each step, the algorithm assigns to an uncolored vertex v one of its possible colors and removes this color from the set of possible colors of v's neighbors. This way, the number of possible colors of another vertex may be reduced to 1; if this is the case, that vertex gets the single possible color and the same rule is used again to further propagate the information. If each vertex has received a color, then a solution has been found and the algorithm terminates. If the set of possible colors of a vertex becomes empty, then the current partial solution cannot be extended to a solution, and so the algorithm backtracks by undoing the last color assignment and trying a new, previously unexplored color. If all possible colors have been tried for a given vertex without success, then the algorithm backtracks one step further. If the algorithm has explored all possible colors of the vertex that was selected first (i.e., it would need to backtrack from the root of the search tree), then the given problem instance is not solvable and the algorithm terminates.

In order to determine which vertex should be colored next, we use the following heuristic. Primarily, we choose the vertex with the smallest number of possible colors because this keeps the number of choices – and thus the size of the search tree – small. If there is a tie, we choose the vertex with the highest number of uncolored neighbors. The rationale is that this choice will lead to a high number of propagation steps, thus narrowing the further search space.

For symmetry breaking purposes, the actual coloring is preceded by a pre-processing phase, in which we find a clique as big as we can (but not necessarily a clique of maximal size because that would be an NP-hard problem on its own). This is done with the following heuristic: for each vertex v, we greedily grow a clique with v as starting vertex. For this purpose, we iteratively extend the clique (which originally consists of v only) with a random vertex that is adjacent to all vertices in the clique, and stop if there is no such vertex. We thus create n cliques, from which we select the biggest. Then the coloring phase starts by pre-coloring the vertices of this clique with the first t colors, where t is the size of the clique. This way, the number of potentially investigated color assignments is divided by $t!$, which is typically a big gain in running time.

The algorithm is implemented using the CLPFD (Constraint Logic Programming over Finite Domains) library of SICStus Prolog [14], version 4.2.3.

4.2.2 Coloring via conversion to SAT

The second algorithm works by converting the list coloring problem to a Boolean satisfiability problem and using an off-the-shelf SAT solver to solve the converted instance.

The main idea of the conversion is to introduce a Boolean variable for each vertex – color pair. That is, the Boolean variable $x_{i,j}$ is true if and only if vertex i is assigned color j. The constraints that each vertex must obtain exactly one color and that adjacent vertices must obtain different colors are easily represented by appropriate Boolean clauses. Symmetry breaking is performed by first finding a clique similarly as in the direct coloring algorithm of Section 4.2.1, and then constraining for each of the vertices in the clique one of the corresponding variables to true. That is, if the clique consists of t vertices, then t unit clauses are added (from which the solver will of course be able to infer the value of several other variables as well using unit propagation). Furthermore, the fact that vertex i may only be colored with colors in L_i is ensured by constraining the other variables of the given vertex to false by means of a unit clause.

Technically, we perform the conversion from list coloring to SAT using SICStus Prolog; the resulting SAT problem is solved using glucose 3.0 [7].

4.2.3 Coloring via conversion to ILP

The third algorithm consists of converting the list coloring problem to an integer linear program (ILP) and using an off-the-shelf ILP solver to solve the converted instance.

The conversion is done in principally the same way as for the conversion to SAT, using the fact that a SAT instance can be easily encoded using integer linear programming [33].

Technically, the conversion from list coloring to ILP is performed using a Java program; the resulting ILP instance is solved using the Gurobi solver[2], version 7.0.

5 Empirical results

In order to assess the impact of the number of colors in the critical range, we conducted a series of experiments, using the three random graph models, the three list coloring algorithms, and different problem sizes. Each data point presented in the following subsections is the average or median of 50 or 100 measurements with the given parameters.

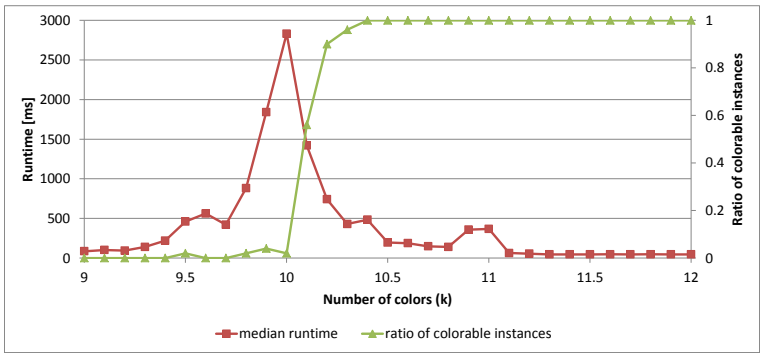
The measurements related to the direct coloring algorithm of Section 4.2.1 were carried out using a laptop computer with Intel Core i5 M520 CPU 2.40 GHz and 3 GB RAM, running Windows 7 Professional 32 bit; those with the SAT-based algorithm of Section 4.2.2 were run on a PC with Intel Core i3-2100 CPU 3.10 GHz and 4GB RAM, running Windows 7 Professional 64 bit; finally, the ILP-based algorithm of Section 4.2.3 was run on a laptop computer with Intel Core i5-4210U CPU 1.70GHz and 8GB RAM, running Windows 10 Enterprise.

5.1 Fractional easy-hard-easy pattern

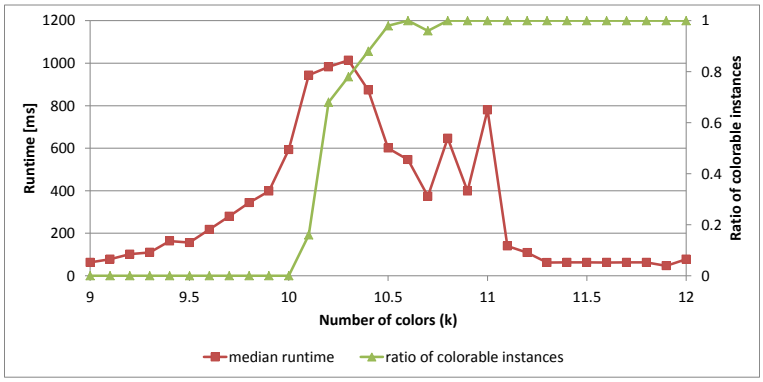
The main finding of our investigations, regardless of the used graph model and list coloring algorithm, is the existence of an easy-hard-easy pattern at the phase boundary also for fractional color numbers, as shown for example in Figures 4(a)-4(c). These figures show three different cases:

- In Figure 4(a), the complexity peaks clearly at 10. On the other hand, the ratio of colorable instances is almost 0 at this point (but 1 at $k = 11$). This means that for almost all of these graphs, $\chi = 11$. In other words, the complexity peaks at $\chi - 1$.
- In Figure 4(c), the complexity peaks clearly at 11. The ratio of colorable instances is almost 1 at this point, but 0 at $k = 10$. This means that for almost all of these graphs, $\chi = 11$. In other words, the complexity peaks at χ.

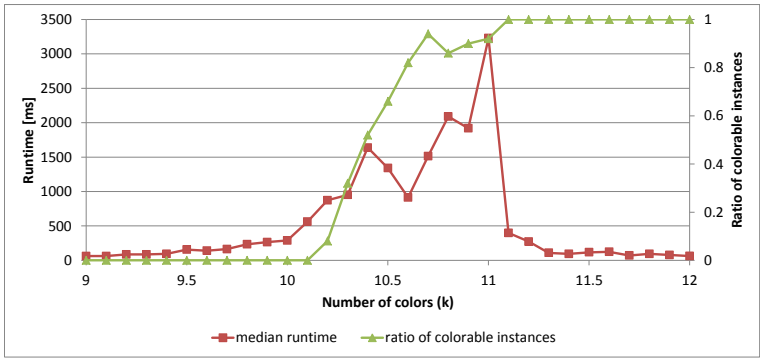
[2] http://www.gurobi.com/



(a) $m = 900$



(b) $m = 920$

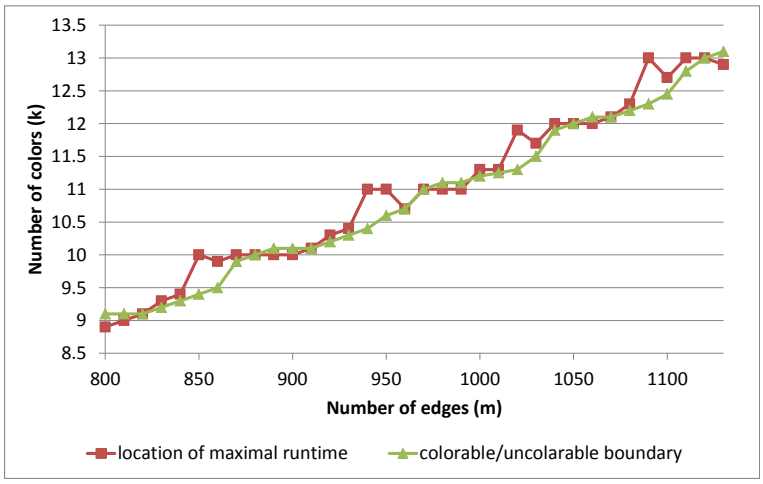


(c) $m = 940$

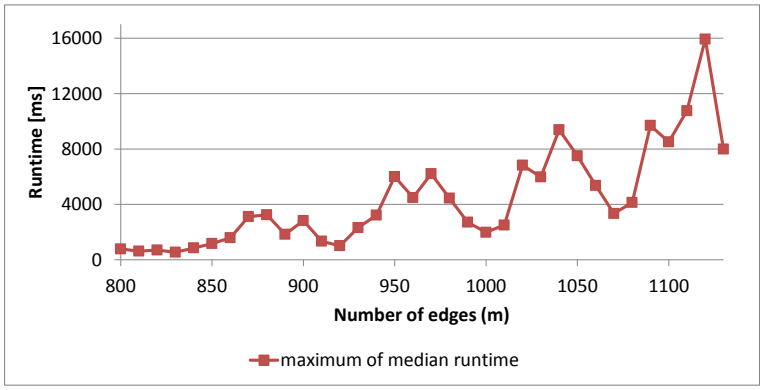
Figure 4: Colorability and complexity as a function of the – fractional – number of colors, for different densities ($n = 60$)

- Figure 4(b) shows a situation between the other two. Also for these graphs, χ is almost always 11. The complexity is quite high in the whole $[10, 11]$ interval, but its maximum is between $\chi - 1$ and χ, namely at $k = 10.3$. The complexity at $\chi - 1$ and χ are comparable. It is also worth mentioning that the maximum value in this case is significantly lower than in the above cases.

The trend exemplified by these figures is quite typical. Also for other values of the parameters, we can observe a similar behavior: when the density of the graphs increases, the peak of the complexity curve also moves to the right; when the maximum is at an integer, this results in a sharp and high peak, whereas if the maximum is between two integers, then the complexity peak is broad and shallow, with also relatively high values at the two integers.



(a) Movement of the maximum's location and the phase boundary



(b) Maximal value of median runtime

Figure 5: Movement of the peak for $n = 60$

In the following, we use this metaphor of the peak's „movement to the right" to describe qualitatively our findings.

5.2 Movement of the peak

In Figure 5(a), we show how increasing the density of the graph shifts both the location of the maximal complexity and the phase boundary (the boundary between the colorable and uncolorable ranges) to the right. More precisely, the two curves show

- the (fractional) number of colors, for which the median runtime is maximal, as a function of the number of edges;
- the (fractional) number of colors, for which the ratio of colorable instances is nearest to 0.5, as a function of the number of edges.

As can be seen in the figure, the two functions are monotonously increasing, except for a couple of small decreases. Moreover, the two curves go mostly together, i.e., the difference of the two functions is quite small.

Also some interesting oscillations can be observed that are not big but since they recur again and again, they may still be significant. In particular, the slope of the curves is relatively low when their y coordinate is near an integer, and higher between the integers. This holds for both curves; the slope of the curve

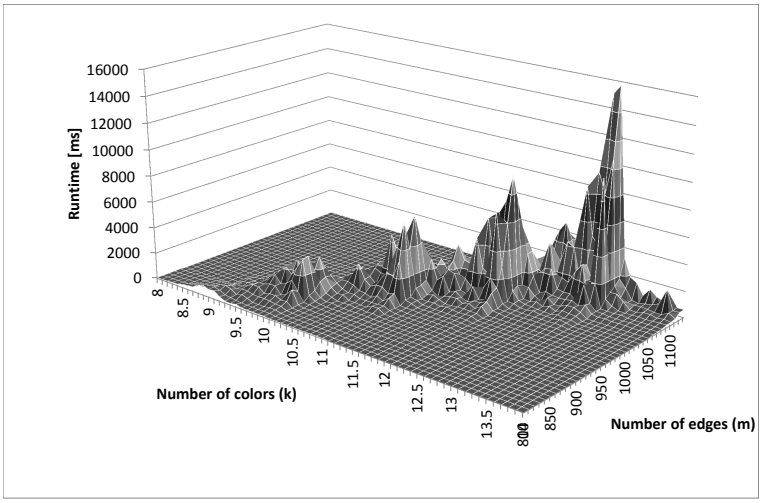


Figure 6: Median runtime as a function of the number of colors and number of edges ($n = 60$)

representing the location of maximal complexity exhibits especially large fluctuations when reaching the next integer. As a result of the difference between the two curves, when the location of the maximal complexity first reaches the next integer ℓ, the phase boundary is still only slightly above $\ell - 1$. These are the points where graph coloring takes longer with χ colors than with $\chi - 1$ colors. For somewhat higher densities though, the phase boundary overhauls the location of maximal complexity, meaning that coloring with $\chi - 1$ colors takes longer than with χ colors.

Figure 5(b) shows the maximal value of the median runtime for different densities in the same range. One can note that this curve also shows „waves", and that these waves are higher and higher. The lack of monotonicity is somewhat surprising, but this is the same phenomenon that was already visible on Figure 4: the maximal value of the median runtime in Figure 4(b) is clearly lower than those in the other two subfigures (recall the different scales on the vertical axis of the subfigures).

It is also interesting to compare Figures 5(a) and 5(b). The local maxima of the curve of Figure 5(b) seem to coincide with those points where the location of the median runtime first reaches the next integer, leaving the phase boundary behind.

5.3 Putting the dimensions together

It is possible to visualize both the dependence on the number of colors and on the number of edges by means of a 3-dimensional plot, as shown in Figure 6. Again, we can observe that, for a fixed density, the median complexity follows an easy-hard-easy pattern, and the maximum complexity goes alternating up and down.

From this 3-dimensional plot, we can also draw new conclusions that are hard to observe in the 2-dimensional cross-sections. We can see that in most of the $m - k$ parameter space, the median complexity is hardly changing and very low, but there is a narrow stripe in which it is much higher and shows great variance. The projections of this stripe on the coordinate axes are straight lines, but as we can see here, the critical stripe is not parallel to any axis, and it does not even follow a straight line in the $m - k$ plane. Rather, it follows a curve in the $m-k$ plane, corresponding to the relationship between m and the chromatic number (cf. Section 2.1).

As we can see in the figure, even when going along the critical curve in the $m - k$ plane, the median complexity is not constantly high, as one may have expected, but oscillates quite wildly.

This picture has important consequences, for example for algorithm benchmarking. First, it shows that in the critical region, complexity is very sensitive to changes in both parameters. Second, if we want to find particularly hard instances in a given size range, we may need to change both m and k to arrive to a local maximum of the runtime as $\mathbb{R}^2 \to \mathbb{R}$ function.

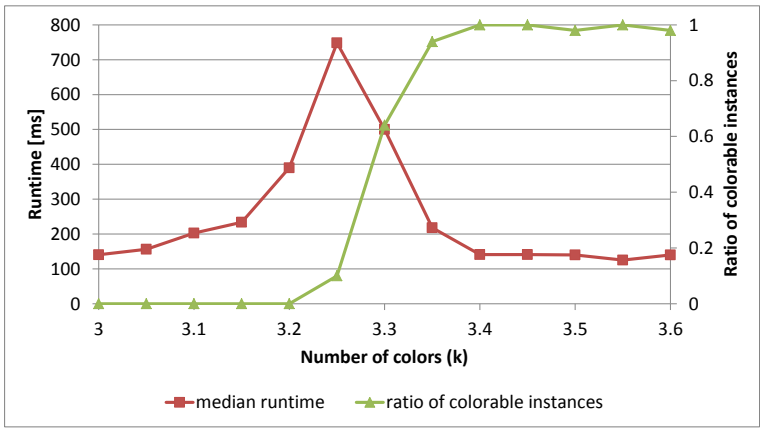


Figure 7: Testing on bigger graphs: $n = 350, m = 1050$

5.4 Variations

Finally, it is important to assess whether our findings also apply to other parameter ranges, graph models, and (list) coloring algorithms.

We conducted several measurements with higher and lower values for the parameters n, m, p, k. A representative example can be seen in Figure 7, which was created with $n = 350$ and $m = 1050$. Qualitatively the same phenomena can be observed in this figure as in the previous ones (e.g., Figure 4). There are some remarkable quantitative differences, though. First, the interval of k values, in which the ratio of colorable instances changes from 0 to 1, is smaller, indicating that the phase transition becomes sharper for bigger graphs. Second, the peak of the median runtime is smaller than before (note the different scale on the left vertical axis). This can be attributed to the relative sparsity of graphs with $n = 350$ and $m = 1050$. Third, the runtime outside the critical region is higher than in the cases of Figure 4, showing that there are components of the runtime that are proportional to n.

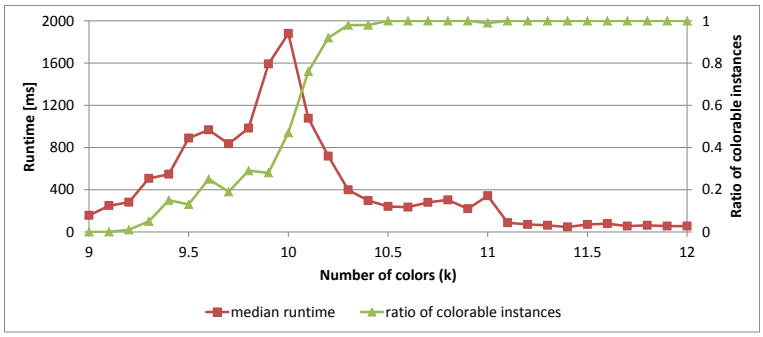


Figure 8: Using the $G_{n,p}$ graph model, $n = 60, p = 0.5$

To assess the impact of the graph model, we also experimented with the $G_{n,p}$ model. An example can be seen in Figure 8. Comparing this figure with the earlier Figure 4(a), the parameters of which are comparable according to formula (1), it can be seen that they are indeed similar. It seems though that plots obtained with the $G_{n,p}$ model tend to be more noisy than the ones obtained with $G_{n,m}$. This can be attributed to the fact that the number of edges of $G_{n,p}$ graphs has some variance around the mean determined by formula (1), whereas $G_{n,m}$ graphs have the same number of edges.

The results of the SAT-based algorithm of Section 4.2.2 are also quite similar to the ones obtained with the direct coloring algorithm of Section 4.2.1. Figure 4(a) was created using the direct coloring algorithm,

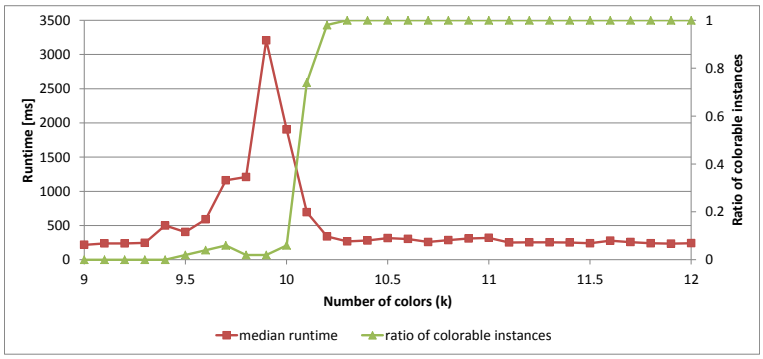


Figure 9: Results of the SAT-based algorithm ($n = 60, m = 900$)

whereas Figure 9 was created with the SAT-based algorithm, with the same parameters. The two figures are indeed very similar. There is one notable difference: the runtime outside the critical region is considerably higher in the case of the SAT-based algorithm, which may be explained by the constant overhead of making the conversion to SAT, writing the resulting SAT problem instance to a file, and invoking the external SAT solver.

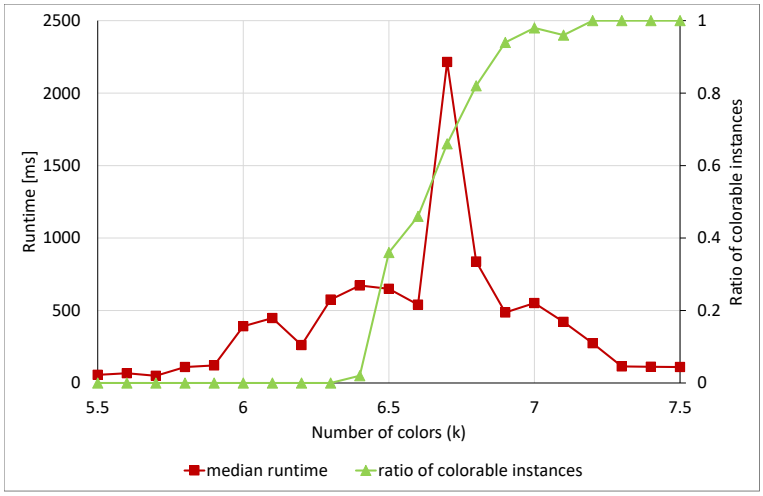


Figure 10: Results of the ILP-based algorithm using the R-MAT graph model ($n = 150, m = 1100, a = 0.35, b = c = 0.25, d = 0.15$)

Finally, Figure 10 shows results measured using the ILP-based algorithm of Section 4.2.3 and using the R-MAT graph model. Because of the realistically clustered structure of R-MAT graphs, deciding their colorability seems to be easier than in the case of the more homogeneous $G_{n,p}$ and $G_{n,m}$ graph models. This is in line with previous experiences showing that deciding colorability of real-world structured graphs is easier than for $G_{n,p}$ and $G_{n,m}$ graphs of similar size [49]. Nevertheless, the pattern seen in Figure 10 is qualitatively very similar to the plots generated with the other graph models and with the other algorithms, showing a clear complexity peak at a fractional k value between $\chi - 1$ and χ.

6 Threats to validity

As with any empirical study, the fact that only a finite subset of the infinite possibilities can be tested poses a threat to the validity of the findings. In particular, it is not possible to study asymptotic properties in a

finite setting. One could, of course, also test larger graphs. The limiting factor is the time needed for the experiments. Given that we are interested in the hardest instances and that the worst-case running time of the investigated algorithms grows exponentially, it is unfortunately not feasible to test on significantly larger graphs, at least not with our currently available infrastructure. Nevertheless, we did test on a range of graph sizes and based on these observations, the presented phenomena seem to be widely applicable.

Another risk is posed by potential tool-specific bias in the measurements. All reported computation times refer to wall-clock time; thus they could be distorted by various technical issues, such as garbage collection or virtual memory page swapping. To mitigate such bias, we used three different coloring algorithms which were implemented using different tools and tested on different machines. To further increase confidence in our findings, we also used three different graph models. The results are largely independent of the used algorithm and graph model.

7 Conclusions and future work

We have shown that list coloring can be used to model graph coloring with a fractional number of colors, and this way we managed to zoom in on the hardest region of the k-colorability problem: where k is at the colorability boundary, between $\chi - 1$ and χ. After a thorough empirical evaluation using three random graph models and three list coloring algorithms, our findings reinforce our initial hypothesis: that the complexity as a function of k exhibits a smooth easy-hard-easy pattern just like it does as a function of the graphs' density.

However, our results revealed much more than that. We observed that with increasing density, the complexity peak moves – together with the phase boundary – to higher k values. The velocity of this movement is not constant but experiences regular fluctuation. In sync with that, also the maximal value of the median runtime shows a periodic wave-like pattern.

Regarding the two independent variables (m and k) and the dependent variable (the median runtime) together in a single three-dimensional plot leads to a fascinating picture. The complexity shows a sequence of sharp local maxima with deep valleys between them. The local maxima follow a well-defined curve on the $m - k$ plane; in other areas of the parameter space, the median runtime is very low. All other plots can be seen as two-dimensional cross-sections of the three-dimensional plots that show different parts of the „whole truth."

With all of the caveats discussed in Section 6, we believe that the presented results give us a better qualitative understanding of the complexity landscape of the k-colorability problem, and may pave the way to improved prediction of algorithm runtime.

An important next step could be to derive a rigorous quantitative analysis of the complexity of backtrack algorithms at the colorability phase transition to mathematically show the corresponding behavior of the algorithms' average-case complexity. Such a theoretical analysis might also make it possible to reason about the asymptotic behavior. Another interesting topic for future research is how real-world graphs, or the typical characteristics of real-world graphs, impact the results.

Acknowledgements

This work was carried out mainly while the author was at Budapest University of Technology and Economics, Department of Computer Science and Information Theory. The work was partially supported by the Hungarian Scientific Research Fund (Grant Nr. OTKA 108947) and the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

[1] Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *49th Annual IEEE Symposium on Foundations of Computer Science*, pages 793–802, 2008.

[2] Dimitris Achlioptas and Michael Molloy. The analysis of a list-coloring algorithm on a random graph. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 204–212, 1997.

[3] Dimitris Achlioptas and Cristopher Moore. Almost all graphs with average degree 4 are 3-colorable. *Journal of Computer and System Sciences*, 67:441–471, 2003.

[4] Dimitris Achlioptas and Assaf Naor. The two possible values of the chromatic number of a random graph. In *36th ACM Symposium on Theory of Computing (STOC '04)*, pages 587–593, 2004.

[5] Noga Alon and Michael Krivelevich. The concentration of the chromatic number of random graphs. *Combinatorica*, 17(3):303–313, 1997.

[6] Péter Arató, Zoltán Ádám Mann, and András Orbán. Time-constrained scheduling of large pipelined datapaths. *Journal of Systems Architecture*, 51(12):665–687, 2005.

[7] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 399–404, 2009.

[8] Edward A. Bender and Herbert S. Wilf. A theoretical analysis of backtracking in the graph coloring problem. *Journal of Algorithms*, 6(2):275–282, 1985.

[9] Béla Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988.

[10] Béla Bollobás. How sharp is the concentration of the chromatic number? *Combinatorics, Probability and Computing*, 13(1):115–117, 2004.

[11] Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[12] Preston Briggs, Keith D. Cooper, and Linda Torczon. Improvements to graph coloring register allocation. *ACM Transactions on Programming Languages and Systems*, 16(3):428–455, 1994.

[13] J. Randall Brown. Chromatic scheduling and the chromatic number problem. *Management Science*, 19(4):456–463, 1972.

[14] Mats Carlsson, Greger Ottosson, and Björn Carlson. An open-ended finite domain constraint solver. In *Programming Languages: Implementations, Logics, and Programs*, pages 191–206, 1997.

[15] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 442–446, 2004.

[16] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In *12th International Joint Conference on Artificial Intelligence (IJCAI '91)*, pages 331–337, 1991.

[17] Simona Cocco and Rémi Monasson. Trajectories in phase diagrams, growth processes and computational complexity: how search algorithms solve the 3-satisfiability problem. *Phys. Rev. Lett.*, 86:1654, 2001.

[18] Joseph Culberson and Ian Gent. Frozen development in graph coloring. *Theoretical Computer Science*, 265(1-2):227–264, 2001.

[19] W. Fernandez de la Vega. On the chromatic number of sparse random graphs. In B. Bollobás, editor, *Graph Theory and Combinatorics*, pages 321–328. Academic Press, 1984.

[20] Liat Ein-Dor and Rémi Monasson. The dynamics of proving uncolourability of large random graphs. I. Symmetric colouring heuristic. *Journal of Physics A: Mathematical and General*, 36:11055–11067, 2003.

[21] Pál Erdős and Alfréd Rényi. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 5:17–61, 1960.

[22] Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57:187–199, 1998.

[23] Michael R. Garey and David S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43–49, 1976.

[24] Carla Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24(1-2):67–100, 2000.

[25] G. R. Grimmett and C. J. H. McDiarmid. On colouring random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 77(2):313–324, 1975.

[26] Francine Herrmann and Alain Hertz. Finding the chromatic number by means of critical graphs. *ACM Journal of Experimental Algorithmics*, 7(10):1–9, 2002.

[27] Tad Hogg. Refining the phase transition in combinatorial search. *Artificial Intelligence*, 81(1-2):127 – 154, 1996.

[28] Tad Hogg and Colin P. Williams. The hardest constraint problems: A double phase transition. *Artificial Intelligence*, 69(1-2):359–377, 1994.

[29] Haixia Jia and Cristopher Moore. How much backtracking does it take to color random graphs? Rigorous results on heavy tails. In *Principles and Practice of Constraint Programming (CP 2004)*, pages 742–746, 2004.

[30] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of computer computations*, pages 85–103. Plenum, 1972.

[31] Tomasz Luczak. The chromatic number of random graphs. *Combinatorica*, 11(1):45–54, 1991.

[32] Tomasz Luczak. A note on the sharp concentration of the chromatic number of random graphs. *Combinatorica*, 11(3):295–297, 1991.

[33] Zoltán Ádám Mann. *Optimization in computer engineering—Theory and applications*. Scientific Research Publishing, 2011.

[34] Zoltán Ádám Mann and Pál András Papp. Predicting algorithmic complexity through structure analysis and compression. *Applied Soft Computing*, 13(8):3582–3596, 2013.

[35] Zoltán Ádám Mann and Anikó Szajkó. Average-case complexity of backtrack search for coloring sparse random graphs. *Journal of Computer and System Sciences*, 79(8):1287–1301, 2013.

[36] Zoltán Ádám Mann and Anikó Szajkó. Determining the expected runtime of exact graph coloring. In *Mini-conference on Applied Theoretical Computer Science (MATCOS), published in the Proceedings of the 13th International Multiconference, Information Society - IS, Volume A*, pages 389–392, 2010.

[37] Zoltán Ádám Mann and Anikó Szajkó. Improved bounds on the complexity of graph coloring. In *Advances in the Theory of Computing (AITC 2010), published in the Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 347–354. IEEE Computer Society, 2010.

[38] Zoltán Ádám Mann and Anikó Szajkó. Complexity of different ILP models of the frequency assignment problem. In Zoltán Ádám Mann, editor, *Linear Programming – New Frontiers in Theory and Applications*, pages 305–326. 2012.

[39] D. W. Matula, G. Marble, and J. D. Isaacson. Graph coloring algorithms. In R. C. Read, editor, *Graph Theory and Computing*, pages 109–122. Academic Press, 1972.

[40] Nirbhay K. Mehta. The application of a graph coloring method to an examination scheduling problem. *Interfaces*, 11(5):57–65, 1981.

[41] Michael Molloy. The freezing threshold for k-colourings of a random graph. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 921–930, 2012.

[42] Rémi Monasson. On the analysis of backtrack procedures for the coloring of random graphs. In E. Ben-Naim, H. Frauenfelder, and Z. Toroczkai, editors, *Complex Networks*, pages 235–254. Springer, 2004.

[43] Rémi Monasson. A generating function method for the average-case analysis of DPLL. In *Proceedings of APPROX-RANDOM '05*, pages 402–413, 2005.

[44] Rémi Monasson, Riccardo Zecchina, Scott Kirkpatrick, Bart Selman, and Lidror Troyansky. Determining computational complexity from characteristic phase transitions. *Nature*, 400:133–137, 1999.

[45] Konstantinos Panagiotou and Angelika Steger. A note on the chromatic number of a dense random graph. *Discrete Mathematics*, 309:3420–3423, 2009.

[46] Bart Selman, David G. Mitchell, and Hector J. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81(1-2):17–29, 1996.

[47] Eli Shamir and Joel Spencer. Sharp concentration of the chromatic number on random graphs $G_{n,p}$. *Combinatorica*, 7(1):121–129, 1987.

[48] Eli Shamir and Eli Upfal. Sequential and distributed graph coloring algorithms with performance analysis in random graph spaces. *Journal of Algorithms*, 5:488–501, 1984.

[49] Tamás Szép and Zoltán Ádám Mann. Graph coloring: the more colors, the better? In *Proceedings of the 11th IEEE International Symposium on Computational Intelligence and Informatics*, pages 119–124, 2010.

[50] Jonathan S. Turner. Almost all k-colorable graphs are easy to color. *Journal of Algorithms*, 9(1):63–82, 1988.

[51] Toby Walsh. The interface between P and NP: COL, XOR, NAE, 1-in-k, and Horn SAT. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 695–700, 2002.

[52] Herbert S. Wilf. Backtrack: an O(1) expected time algorithm for the graph coloring problem. *Information Processing Letters*, 18:119–121, 1984.

[53] Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1173–1178, 2003.

[54] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.