Frozen Development in Graph Coloring

Joseph Culberson [1]

Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, T6G 2H1. joe@cs.ualberta.ca

Ian Gent [1]

School of Computer Science, University of St Andrews, St Andrews, Fife KY16 9SS, United Kingdom. ipg@dcs.st-and.ac.uk

Abstract

We define the 'frozen development' of coloring random graphs. We identify two nodes in a graph as frozen if they are the same color in all legal colorings and define the collapsed graph as the one in which all frozen pairs are merged. This is analogous to studies of the development of a backbone or spine in SAT (the Satisfiability problem). We first describe in detail the algorithmic techniques used to study frozen development. We present strong empirical evidence that freezing in 3-coloring is sudden. A single edge typically causes the size of the graph to collapse in size by 28%. We also use the frozen development to calculate unbiased estimates of probability of colorability in random graphs. This applies even where this probability is infinitesimal such as 10^{-300}, although our estimates might be subject to very high variance. We investigate the links between frozen development and the solution cost of graph coloring. In SAT, a discontinuity in the order parameter has been correlated with the hardness of SAT instances, and our data for coloring is suggestive of an asymptotic discontinuity. The uncolorability threshold is known to give rise to hard test instances for graph-coloring. We present empirical evidence that the cost of coloring threshold graphs grows exponentially, when using either a specialist coloring program, or encoding into SAT, or even when using the best of both techniques. Hard instances seem to appear over an increasing range of graph connectivity as graph size increases. We give theoretical and empirical evidence to show that the size of the smallest uncolorable subgraphs of threshold graphs become large as the number of nodes in graphs increases. Finally, we discuss some of the issues involved in applying our work to the statistical mechanics analysis of coloring.

[1] Joseph Culberson's work was supported in part by an NSERC grant OGP8053 and an EPSRC visiting fellowship, GR/M54605. We thank Riccardo Zecchina for many helpful discussions, and David Wilson, David Mitchell, and Paul Beame for helpful discussions at the Trieste meeting. We are members of the APES Research

1 Introduction

A phase transition has been identified for many NP-complete problems and is frequently correlated with a high frequency of hard instances. This contrasts with randomly chosen instances from other regions of the problem space, where most often such instances are easy. Recently the techniques of statistical mechanics have been applied to the analysis of this transition, and have yielded insights into the nature of the region and its relation to hardness. (See [12] for an overview and many references).

One of the more recent efforts has been the identification of the nature of the order parameter at the transition. In some problems (k-SAT, $k > 2$) there is evidence that this parameter is discontinuous [19,21,22]. These problems typically have a high frequency of hard instances. On the other hand problems with a continuous parameter tend not to have a high frequency of hard instances. This has been explored most thoroughly in the case of $2 + p$-SAT [20,21]. For the Hamiltonian Cycle problem, the order parameter may also be continuous [30] which corresponds to a very low frequency of hard instances at the transition [28].

For SAT one measure of the order parameter is the *backbone* [21] which is the number of variables that are frozen to a particular value under all satisfying assignments. A closely related notion is the *spine* [4]. For k-SAT, $k > 2$, the evidence indicates that this measure jumps from zero to a fixed fraction of the n variables at the transition. Parkes showed that at the satisfiability threshold, many variables are frozen although some are almost free [24]. Similarly, for Hamiltonian Cycle we can use as variables the number of edges that must appear in all Hamiltonian Cycles of the graph. In contrast to the K-SAT case, only a few edges are frozen in satisfiable instances at the transition.

For k-coloring (we consider $k = 3$ almost exclusively) it is more difficult to define and measure the order parameter, due in large part to the symmetry of the solutions. However, it is clear that if we add an edge to a colorable graph and it then becomes uncolorable, then in the graph without the edge the pair of vertices must have been colored with the same color in every coloring of the graph. Such a pair we refer to as frozen for reasons that will become apparent in the next section. We adopt the number of such pairs as our spine, an approximation to the order parameter for coloring, as it mimics closely the backbone and spine used in other problems. Our spine shows strong evidence of a discontinuous jump at the transition. We discuss some of the possible consequences for the statistical mechanics analysis of coloring, in particular on the likelihood of discontinuous behavior in possible order parameters.

Group, `www.apes.cs.strath.ac.uk`, and we thank other members for their help with this work, especially Isla Ross, Josh Singer and Toby Walsh.

Instances at or near thresholds are used for benchmarking algorithms in many domains, for example satisfiability [18] and constraint satisfaction problems [25,27]. However there is no simple correlation between hard instances and phase transitions. For example, there is a classic phase transition in the solvability of random Hamiltonian cycle instances [15] but hard instances do not seem to be found there [28]. We provide extensive evidence that 3-coloring problems are hard at the phase transition and beyond. We attempt to understand why phase transitions so often yield hard instances. We believe, and the evidence supports, that the reason for the exponential growth is the disappearance of small $k+1$-critical subgraphs at the threshold. There is no known way of efficiently noticing and verifying the existence of large critical subgraphs of non-k-colorable graphs.

1.1 Overview of the paper

In Section 2.1 we present the model, which we call the full frozen development, used to study the properties of the phase transition. Naive methods for studying frozen development would be prohibitively costly, but in Section 2.2 we present a detailed overview of the techniques we use to calculate frozen development. In theory these allow the computation to be performed using $O(n^2 \log(n))$ calls to a graph coloring program, and in practice they allow us to do the complete computation for n up to 225,

In Section 3 we present the empirical results of the study on our frozen development model, showing strong evidence that indeed our backbone measures show a discontinuous jump at the threshold. In particular we show evidence that a fixed fraction of pairs become frozen under the addition of $O(1)$ edges. We show in Section 3.2 that this jump can be seen as a sudden collapse in the size of the graph. If we consider the graph induced by the equivalence relation defined by frozen pairs, the addition of a single edge typically reduces the size of the graph by 28%. As a final application of frozen development, we report in Section 3.3 on new estimates on the probability of colorability into regions where this probability can be as low as 10^{-300}.

In Section 4 we present empirical evidence, using two distinct complete algorithms, to show that indeed there is exponential growth in difficulty near the threshold as n increases. This evidence is a mixture of results measuring difficulty exactly at the threshold graph from the frozen development process, and of further graphs generated from the G_{np} model for larger values of n.

In Section 5 we present theoretical and empirical evidence that the smallest 4-critical subgraphs of a non-3-colorable graph are large near the threshold asymptotically. We correlate this with the hardness of instances empirically observed. We wonder whether a discontinuity in a backbone is suggestive of such large critical structures in general.

In the conclusions (Section 7) we discuss the possibility that the discontinuity exhibited by our measure may not correspond to a discontinuous order parameter as measured by a minimization of violated edges over all partially correct 3-colorings. If this should prove to be the case, it could have implications for refinements needed in the analysis as applied to this problem.

2 Frozen Development and Algorithmics

2.1 Full Frozen Development

For purposes of this study, we consider a model which can be viewed as an implementation of the standard random graph process as described in [3, Chapter VII]. It is based on the set of $\binom{n}{2}$ unordered pairs of distinct vertices $\{\{u,v\}, u,v \in \mathbf{V}\}$ where $n = |V|$. We generate at random one of the $\binom{n}{2}!$ permutations of the set of pairs, calling this the *input sequence* π. We build a graph of m edges by choosing the first m pairs from the input sequence. For a given sequence π we can determine the smallest value $t(\pi)$ (or t when π is understood) such that the graph on the first $t(\pi)$ edges is not k-colorable: we call $t(\pi)$ the *threshold index of the sequence* π. We define the *average threshold* $T(n)$ as the average over the set of all input sequences Π of the values t.

$$T(n) = \frac{1}{\binom{n}{2}!} \sum_{\pi \in \Pi} t(\pi)$$

One of the conjectures on the coloring phase transition says that for each k, $T(n)/n$ converges to a constant α_k as $n \to \infty$.

In practice we do not want to stop measurement at the threshold index; after all if some other edge had occurred at this point in the sequence we might be able to keep on going. In the *full frozen development method* when a pair π_m is added as an edge that renders the graph non-k-colorable, then the edge is deleted and we move on to the next edge. This gives us a way to smoothly extend frozen measures beyond the threshold until all pairs have been considered. At that point, the graph is a maximal k-colorable graph.

To make this more precise we first present a few definitions.

Given a k-colorable graph G, we say that a pair $\{u, v\}$ is *frozen* iff for every coloring c of G, $c[u] = c[v]$. The name frozen means that this edge cannot be in any k-colorable graph containing G as a subgraph.

The *spine* (or backbone)[2] of a colorable graph G is

$$\mathbf{B}(\mathrm{G}) = \{\{u,v\} : \{u,v\} \text{ is frozen by } \mathrm{G}\}$$

We define the sequence of graphs $\mathrm{G}_i = \mathrm{G}_i(\pi), 0 \le i \le \binom{n}{2}$, inductively.

$$\mathrm{G}_0 = (\mathbf{V}, \emptyset)$$
$$\mathrm{G}_i = \begin{cases} \mathrm{G}_{i-1}, & \pi_i \in \mathbf{B}(\mathrm{G}_{i-1}) \\ \mathrm{G}_{i-1} + \pi_i, & \text{otherwise} \end{cases}$$

We follow Bollobás *et al* [4] by defining the *scapegoats* of this sequence as those pairs π_i which are not included because they occur in the spine of an earlier graph. The name arises because each scapegoat could be held responsible for uncolorability if it were added to the graph. That is, the set of scapegoats for a sequence π is

$$\mathbf{S}(\pi) = \{\pi_i : \pi_i \notin \mathrm{G}_i(\pi)\}.$$

Note that the first scapegoat encountered is the threshold for the sequence. For $m < t$ the graph G_m is the graph formed by the first m pairs of the sequence. Notice that $\mathrm{G}_{\binom{n}{2}}$ is a k-partite complete graph.

For each G_m we record all those pairs which are frozen by G_m. That is, each remaining pair $\pi_i, i > m$ is tested and recorded as frozen with *crystallization index* $\iota(\pi_i) = m$ iff π_i is frozen by G_m and not by G_{m-1}. Notice that the threshold index t is exactly the smallest index m such that π_m is frozen for some index $\iota(\pi_m) < m$. Also, we see that $\mathbf{B}(\mathrm{G}_m) = \{\pi_i : \pi_i \in \mathbf{S}(\pi), \iota(\pi_i) \le m\}$.

We can also compute the smallest value m^f of m such that there is some pair (u,v) of vertices (necessarily with index in the sequence greater than m^f) frozen the same. There may be more than one pair which are forced to be the same color by the addition of the m^fth edge. We call π_{m^f} the first edge causing a frozen pair, or the *first frozen* for short. We define

$$F(n) = \frac{1}{\binom{n}{2}!} \sum_{\pi \in \Pi} m^f(\pi)$$

[2] We choose the term spine because in the full frozen development it most closely approximates the spine defined in [4]. While not identical, it shares the important property of monotonicity with the earlier definition.

Clearly, for every sequence π, $m^f(\pi) \leq t(\pi)$. Thus, $\gamma_k = \lim_{n\to\infty} F(n)/n$ (assuming it exists) must be bounded above by α_k. Elsewhere we have presented evidence to strongly suggest that as $n \to \infty$, $\gamma_3 \to \alpha_3$ [8].

Analogously to frozen pairs, we can measure for each m the number of pairs $\{u, v\} = \pi_i, i > m$ such that for all colorings of G_m, $c[u] \neq c[v]$. Such a pair is said to be *free.* We observe that free is not quite the complement of frozen. When a pair is frozen it cannot appear in a k-colorable graph which is a super graph of the current one. When a pair is free its addition makes no difference to the set of legal colorings of any super-graph. Thus, in calculating frozen development we can skip adding these edges, because they simply do not matter to the set of colorings.[3] As for frozen pairs, the free pairs are also part of the crystallization process. We associate with each free pair the crystallization index when it first became free.

Our program for the full frozen development process on a sequence π produces as output the type of each pair $\pi_m, 1 \leq m \leq \binom{n}{2}$. This type is either frozen, free or *effective*. Effective means that for all $i < m$, π_m is neither frozen nor free in G_i. Thus the edge is effective in the sense that it is added to the sequence of graphs without causing uncolorability, and it does reduce the number of colorings of G_m to less than those of G_{m-1}.[4]

For each non-effective pair, we record the index at which it first becomes frozen or free with the index of effective pairs being set to themselves. With this information, it is easy to compute the growth of the spine, and of the free set, as well as other statistics such as the location of the first frozen and the threshold.

Our model and measure of frozen pairs is not the same as the order parameter used in statistical physics. To study the order parameter empirically, we might generate random graphs usig the G_{np} model, where each pair of vertices is an edge with probability p. To measure the order parameter we would need to obtain all colorings which violate the minimum number of edges.

Our model gives us several advantages in studying the phase transition over this alternative. Recall that for convex properties (such as colorabillity) results converge asymptotically for G_{np} and G_{nm} models provided $p\binom{n}{2} \sim m$ [3,23]. One advantage of our model is an expected reduction in variance on computing $T(n)$ for a given sample size. Thus we obtain a better estimate of the threshold at a cheaper computational cost.

[3] When we test the hardness of coloring, all free edges must be added. The free edges could make a big difference to a coloring algorithm, for example by preventing errors high in the search tree.

[4] Note that our definitions ensure that effective edges are *not* free, even though (trivially) $c[u] \neq c[v]$ in G_m if $\pi_m = \{u, v\}$.

A second advantage is that the computing costs for our notion of frozen sets is less than for the order parameter or backbone. In the next section we give details of our implementation together with some efficiency analysis. We see no way of obtaining this level of efficiency for computing all minimum violation colorings, or even the non-monotonic backbone as used in [19,20] and elsewhere. For random non-colorable graphs there could be many different subsets of edges of minimum size that could be violated. Just determining these subsets is already difficult.

Finally, in section 6 we discuss the possibility that while our measure shows clearly a sudden jump in frozen pairs, it is possible that a measure based on minimum violation colorings may not so clearly show such a jump. It could be especially difficult to find a jump in an empirical study which must necessarily be on examples of small size. Since we tend to believe that such sudden jumps are correlated with the structures that create hard instances, the alternative could prove to be less informative as an indicator of hardness.

2.2 Efficient Computation of Full Frozen Development

We turn our attention to the outline of the method we use to compute the full frozen development. The reader not interested in these details but only in the empirical results can safely skip to the next section.

Our approach can be broadly classed as a forward scanning method since for each G_m we create, we scan forward over remaining pairs to determine whether each is free, frozen or as yet not crystallized. We outline several optimizations that make this task amenable up to $n = 225$. As a result, empirical indications are that this runs in about $O(n^2 \log n)$ calls to our underlying coloring algorithm. At the end of this section we show that the full frozen development can indeed be performed in at most $O(n^2 \log n)$ calls.

To determine whether a pair $\pi_i = \{u, v\}$ is frozen at index m, we test if the graph $\mathrm{G}_m + \{u, v\}$ is k-colorable. If not, the pair is frozen.

To test whether the pair $\pi_i = \{u, v\}$ is free, we merge u to v. A merge requires v to be deleted and then for every edge $\{v, z\}$ that was in G_m, we add an edge $\{u, z\}$ to $\mathrm{G}_m \circ (uv)$ if it is not already present. If $\mathrm{G}_m \circ (uv)$ is not k-colorable then the pair $\{u, v\}$ is free.

These two tests are helpful because they replace an examination of all colorings to single calls to a graph coloring program. This coloring program can be one independently developed and highly optimized, and is freed from the space demand of storing all colorings. The coloring program we use is Smallk which is briefly outlined in appendix II.

The first step in our process is finding the threshold of the sequence. For some purposes, for example when n is much greater than 225, this is the only step we perform. To find the threshold index t, we do a binary search to find the smallest i such that the graph with the edge set $E = \{\pi_1, \ldots, \pi_i\}$ is not k-colorable and set $t(\pi) = i$. This requires $O(\log n)$ calls to the coloring program.

To compute the full frozen development using a naive implementation, we would call a coloring program for each $\mathrm{G}_m, m = 1 \ldots \binom{n}{2}$ and for each pair $\{u, v\}$ later in the sequence π. This would result in $\binom{\binom{n}{2}}{2} = O(n^4)$ calls. Instead, we have taken advantage of the monotonicity of graph colorings to greatly reduce this number. The first family of properties we use to reduce calls are straightforward:

- $\{u, v\}$ free (respectively *frozen*) at index $m \implies \{u, v\}$ not frozen (*not free*) at index m.
- $\{u, v\}$ free (*frozen*) at index $m \implies \{u, v\}$ free (*frozen*) at index $i > m$.
- $\pi_m = \{u, v\}$ frozen at index $m-1 \implies \mathrm{G}_{m-1} + \{u, v\}$ is uncolorable so we can skip π_m.
- $\pi_m = \{u, v\}$ free at index $m-1 \implies$ the set of edges free (*frozen*) at index $m =$ the set of edges free (*frozen*) at index $m-1$. Again we can skip π_m.

We can make further improvements by making use of the coloring found by our coloring program when the graph given to it is colorable. In particular, we can greatly reduce the number of calls to the coloring program while calculating the frozen pairs at a particular index m. We do this by making use of every coloring found, whether when adding edges or merging nodes. We have the following facts:

- $c[u] = c[v] \implies \{u, v\}$ not free at index m.
- $c[u] \neq c[v] \implies \{u, v\}$ not frozen at index m.
- any coloring c' for $\mathrm{G}_m + \{u, v\}$ is also a coloring of G_m.
- any coloring c' for $\mathrm{G}_m \circ (uv)$ can be extended to a coloring for G_m by setting $c'[v] = c'[u]$.

These facts are more useful than they might appear. When we need to test G_m, i.e. we are not skipping pair π_m, we test the first pair $\pi_i = \{u, v\}, i > m$ whose status is not known from smaller indices. If $\mathrm{G}_m + \{u, v\}$ is uncolorable, we have determined that the pair is frozen. If however it is colorable, we can use the coloring c_1 and the first two facts above to rule out one of the possibilities for each $\pi_j, j > i$. If the merge $\mathrm{G}_m \circ (uv)$ gives another colorable graph (in which case π_m is known to be effective) we use the new coloring c_2. For any future pair $\{z, w\}$ such that $c_1[z] \neq c_1[w]$ but $c_2[z] = c_2[w]$, or vice versa, $\{z, w\}$ is effective since it can be neither free nor frozen at index m, meaning that we need not use our coloring program to test either $\mathrm{G}_m + \{z, w\}$ or $\mathrm{G}_m \circ (zw)$.

Typically we will find many such pairs, greatly reducing the number of calls.

The preceding paragraph gives the information we can glean from the first pair $\{u, v\}$ neither free nor frozen at index m. The minimum information we have about each future pair is that only one possibility of free or frozen remains. This cuts the number of calls by at least half. As we test each unresolved pair $\{z, w\}$ for frozenness, the result will give us significant amounts of information. If the pair is frozen, the pair is frozen for all indices $i > m$. If the pair is not frozen, we obtain a new coloring. If we are testing $\{z, w\}$ for being frozen, all previous colorings must have $c[z] = c[w]$ but the new coloring must have $c[z] \neq c[w]$: the case for free is similar. In either case, the new coloring must be different from all previous colorings found for G_m. Barring rare cases, this will determine some future pairs as being neither free nor frozen.

These optimizations give a dramatic reduction in the number of coloring calls necessary. In fact, almost every coloring attempt gives us significant amounts of information: either that a certain pair is free or frozen for many indices, or that many pairs are neither free nor frozen at a certain index. We have still further advantages from the well known easy-hard-easy pattern of coloring, with hard instances concentrated at the threshold. While we do have to perform many coloring calls near the threshold, a large number of calls are either significantly before or after the threshold: these calls are usually cheap.

Finding the threshold as the first step also gives another advantage. Typically the threshold occurs just after a large jump in the number of frozen pairs, as we will see in the next section. A coloring is an equivalence relation, and so is frozen. If $\{u, v\}$ and $\{v, w\}$ are frozen, then taking the transitive closure shows that $\{u, w\}$ is also a frozen pair, and this without a call to the coloring program. On the first forward scan from the threshold we typically find many frozen pairs, and so many pairs never need be checked. The sharper the rise in the frozen pairs, the more effective this is.

The net result is that we can test full frozen development in 3-coloring up to graphs with 225 nodes.

Despite major advantages, our approach may not be optimal. Instead of doing the forward scan, we can use a backward scanning approach. After using a binary search to find $t(\pi)$ we know that all pairs $\pi_m, m < t$ are effective or free, and so require no further testing for frozen. Inductively, after we have determined the values for the sequence up to $m-1$, we test $\mathrm{G}_{m-1} + \pi_m$. If this is not colorable, then π_m is frozen. We may then do a binary search on the set of graphs $\mathrm{G}_1, \ldots \mathrm{G}_{m-1}$, adding π_m to each in turn, to determine $\iota(\pi_m)$. On the other hand, if G_m is not frozen, we can merge the pair of vertices in π_m to see if it is free. If so, we can do a binary search on the graphs $\mathrm{G}_1 \ldots \mathrm{G}_{m-1}$, with the merge on π_m applied to each in turn, to determine $\iota(\pi_m)$ for this free edge.

Otherwise the edge is effective, and requires no further action. This method will do at most $O(n^2 \log n)$ calls to the coloring algorithm. As with the forward scanning method, many of the potential calls can be eliminated by keeping track of certain colorings, and taking transitive closures. We leave practical investigation of that technique for future work. The forward scanning method is more effective for computing certain partial frozen development information.

3 Empirical Evidence for an Asymptotic Discontinuity

We now report empirical results of full frozen development on 3-coloring, using the algorithmic techniques developed above. We will show dramatic evidence strongly suggestive of an asymptotic discontinuity in behavior. Further research is necessary to confirm whether this discontinuity will be apparent in the order parameter for graph coloring.

3.1 Sudden Freezing in 3-coloring

Our first empirical observation is that the freezing process is not gradual. Instead, we see that a large number of pairs freeze at the same index. That is, the addition of one edge can cause many pairs to become frozen simultaneously. Typically, just before the threshold, there is a sequence of from 1 to 5 edges which cause on average 16% or more of the pairs to be frozen. This behavior is typical of the big jump we might expect if there is an asymptotic discontinuity in behavior.

In figure 1 we show how the number of frozen and free pairs grows as the edge density increases. We ran the full frozen development on 100 random permutations at each value of $n = 50, 75, \ldots, 225$ in steps of 25. This is clearly typical of a phase transition, and the sharpness suggests that there will be a discontinuity at $n = \infty$. The range over which these curves are plotted is the entire range over which any difference in the number of frozen or free pairs occurred. For the larger values of n it is of course a tiny fraction of the set of $\binom{n}{2}$ pairs.

In table 1 we present data on the size of the spine at the threshold, and as one might expect the threshold typically occurs after the big jump, although occasionally it occurs by sheer accident after only one or two pairs are frozen.

The spine is monotonic, like the spine of Borgs et al [4]. That is, once a pair is in the spine it remains there as new edges are added. This is different from the backbone of Monasson et al [22], defined with respect to optimizing

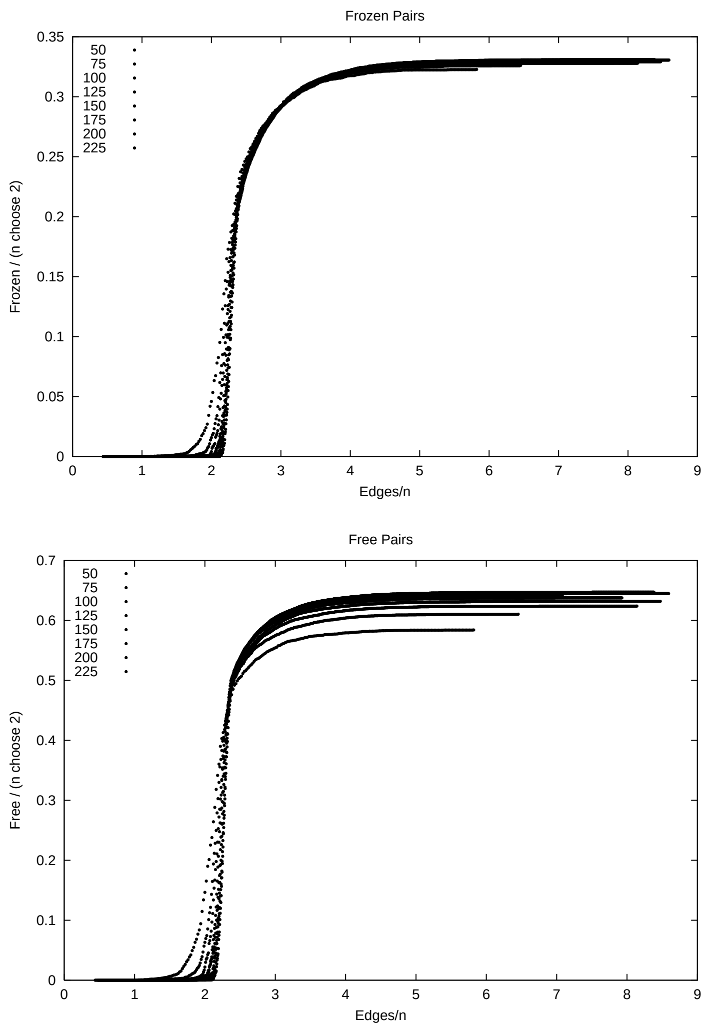


Fig. 1. The ratio of frozen pairs and free pairs to $\binom{n}{2}$ plotted against the ratio m/n.

assignments. The closest analogy in coloring would be a measure with respect to all colorings minimizing the number of violated edges. A frozen pair in a colorable graph might become unfrozen after the threshold where no coloring satisfies all edges, if some optimal colorings violate the edge and others don't. We prefer the monotonic measure we use, not least because it makes possible the $O(n^2 \log(n))$ procedure reported in Section 2.2, but it is possible that less dramatic behavior would be observed with a backbone-like measure. We discuss this issue further in the conclusions.

3.2 Collapse in Graph Structure

We have already noted that frozen defines an equivalence relation on G. We write $\mathcal{F}$ for this relation, so that $u \equiv_{\mathcal{F}} v$ iff the pair $\{u, v\}$ is frozen in G.

Frozen Pairs at the Threshold					
N	Avg	Std	Avg/$\binom{n}{2}$	Min	Max
50	178.88	95.17	0.15	1	347
75	431.55	205.42	0.16	24	760
100	703.41	335.94	0.14	26	1339
125	1154.86	559.06	0.15	1	2001
150	1648.85	750.45	0.15	2	2797
175	2113.37	1082.65	0.14	1	3779
200	3010.74	1283.94	0.15	19	5255
225	3646.12	1606.82	0.14	8	6321

Table 1
The number of frozen pairs at the threshold. Notice that the number is on average about 14% of the total pairs, which can be compared to figure 1

This gives us the natural concept of the graph induced by the equivalence relation, which we denote $G/\mathcal{F}$ and call the *collapsed graph.* If $G = (\mathbf{V}, \mathbf{E})$, we define $G/\mathcal{F} = (\{\|v\| : v \in \mathbf{V}\}, \{\{\|u\|, \|v\|\} : \{u, v\} \in \mathbf{E}\})$, where $\|u\|$ is the equivalence class of u. By definition, for every pair of vertices in the collapsed graph not joined by an edge there is a k-coloring of the graph that makes the vertices different colors. That is, there are no frozen pairs in the collapsed graph. There may, however, be free pairs of vertices.

The collapsed graph gives us an alternative view of freezing. It addresses a potential objection, that the big jump in freezing may occur only due to transitivity. That is, a large number of frozen pairs may occur only because two equivalence classes merge. We will show that the collapsed graph shows dramatic behavior, as did the raw number of frozen pairs. This cannot be solely because of large equivalence classes merging, since that would only entail a reduction of one in the size of the collapsed graph.

We can analyze the nature of the collapsed graphs using the same frozen developments reported in figure 1. We report the sequence of the number of vertices in the collapsed graph for 3-coloring examples. That is, we report the number of vertices in $G_m/\mathcal{F}$ for each m. Notice that we do not add edges to the collapsed graph: we calculate the frozen development as before and then the size of the collapsed graph. In figure 2 we show the collapse in a sample of fifteen 200-vertex graphs, graphs 1-15 of our sample of 100. In each case there is indeed a sudden collapse near the threshold in each sequence. Each of these instances drops rapidly somewhere in the range 430 to 470, which when divided by 200 gives a ratio of 2.15 to 2.35 as we might expect. The threshold graph typically occurs very shortly after the big drop given that there are so many frozen pairs lying around after the big drop (the threshold pair must be

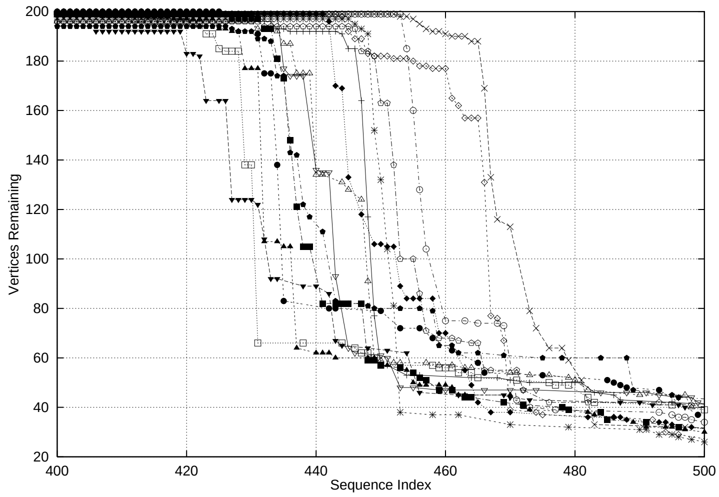


Fig. 2. Collapse in fifteen 200 vertex instances. The y-axis represents the number of vertices remaining in the collapsed graph. The x-labels indicate that the x-axis is the sequence index, that is the number of edges that would be added if none were skipped.

a frozen pair). The marked points on the curves represent the *effective* edges, that is those edges that were neither frozen nor free in the sequence when encountered and so were actually added to the graph. The number of effective edges actually required to cause the catastrophic drop is even smaller than the range of sequence indices indicates.

If we compute the average number of vertices in the collapsed graph over the sample set, we no longer see a sudden drop. In this respect, the average is not a good indicator of what is happening in the individual instances. To show this, in figure 3 we plot the average number of vertices remaining in the collapsed graph for each index, together with another set of twenty 200-vertex graphs. The average is taken over all 100 samples, and the individual graphs are 41-60 from our sample. Almost all instances exhibit a very narrow range over which they drop from a fairly well defined region at the top to another at the bottom. And all have at least one large drop caused by a single edge. The average really only reflects the percentage of instances that have dropped so far, not how fast they drop.

We can use alternative measures to see how fast the size of collapsed graphs drop on individual permutations. To do so we find, for each permutation's full frozen development, the one edge that caused the maximum drop in the number of vertices on that sequence. Call this the *maxdrop* of the experiment. We can now study each permutation's behavior relative to its own maxdrop. We show this in figure 4. As before the y-axis is the average size of collapsed graph, but the x-axis is now the number of edges in the sequence before or after the maxdrop, with the maxdrop edge index represented by $x = 0$. We see from figure 4 that the mean maxdrop is from 72% to 44%, a drop of 28%

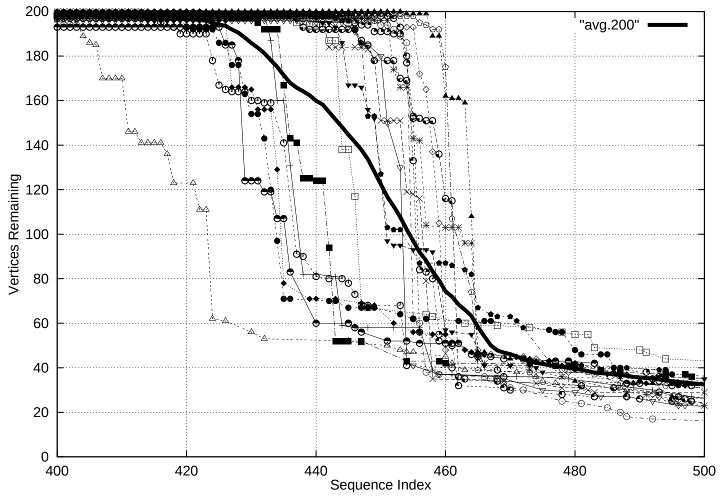


Fig. 3. A set of twenty sample collapses with $n = 200$ with the average of the entire 100 samples overlaid. Notice that the average does not closely approximate the shape of any of the curves, even the most atypical case at the far left.

of the vertices on a single edge. This is represented by a near-vertical jump in the average curve.

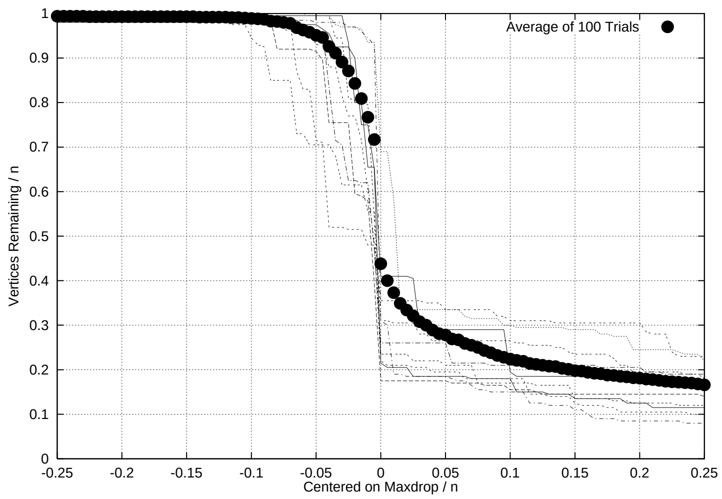


Fig. 4. The first ten experiments from figure 2 and the average of all 100 experiments, but with the samples and average aligned at the index of the maxdrop edge.

The behavior of the maxdrop is remarkably consistent as we change n. In figure 5 we plot in the form of error bars for each n the mean, minimum, and maximum maxdrop of the set of samples. Even the minimum is consistent, with the smallest fraction at 10% for $n = 50$. That is, in every one of the 800 trials, there was a single edge that caused a fractional drop of at least 10% in the size of the collapsed graph. The average of $\approx 28\%$ is very consistent. Figure 5 also shows statistics of the size of the collapsed graph when the maxdrop occurs. We show, in figure 6 the average location of the maxdrop edge. As expected the average index of the maxdrop edge appears to be converging towards 2.3.

Another way of measuring the graph collapse is to compute intervals over which certain drops in the size of collapsed graphs occur. In figure 6 we plot the index (as a ratio to n) where the collapsed graph first has fewer than 80% of its original vertices. In contrast, in figure 7 we plot the actual number of edges required to cause the graph to drop from $0.9n$ down to $0.4n$ vertices. This 50% interval seems not only to be converging relative to n but even in absolute terms. In fact, there are specific instances where an edge caused the drop from above 90% to below 40%, causing the measured number of edges to be zero.

We find it intuitively surprising that a single edge would cause a fixed fraction of vertices to collapse. However, our empirical evidence strongly suggests that this occurs in almost every instance. Accordingly we conjecture that it will be true in the limit. Further theoretical research is necessary to confirm or refute this conjecture.

Our conjecture is highly suggestive of an asymptotic discontinuity in an order parameter, as previously seen in SAT. However, on this point we must be less definite, because it is not yet known what the order parameter for coloring is. Further, as we have seen, some measures of graph collapse (such as the mean size of graphs) do not seem to have a discontinuity. The empirical evidence of sudden collapse in graph is certainly worthy of further investigation from a statistical mechanics viewpoint.

3.3 New Estimates of Probability of Colorability

As well as investigating the region where threshold graphs occur, we show that frozen development is also useful for gaining a more accurate picture of the phase transition in colorability. The main advantage is that we can get unbiased estimates of the probability in colorability both where the probability is very close to one and where it is very close to zero. Additionally we see a variance reduction compared to simple generate and test, but this reduction does not repay the computational expense of the method.

Recall that in frozen development we have a permutation π of all $\binom{n}{2}$ pairs of vertices. Also, recall that an edge π_i is a scapegoat exactly if the edge π_i is not in $\mathrm{G}_i(\pi)$. The first scapegoat occurs at the threshold, and thereafter we see more scapegoats. Any sequence π can be *reduced* to a scapegoat-free subsequence $R(\pi) = \pi - \mathbf{S}(\pi)$, defined simply as the set of non-scapegoats in π in the same order in which they occur in π. Thus, $R(\pi)$ defines the edges in

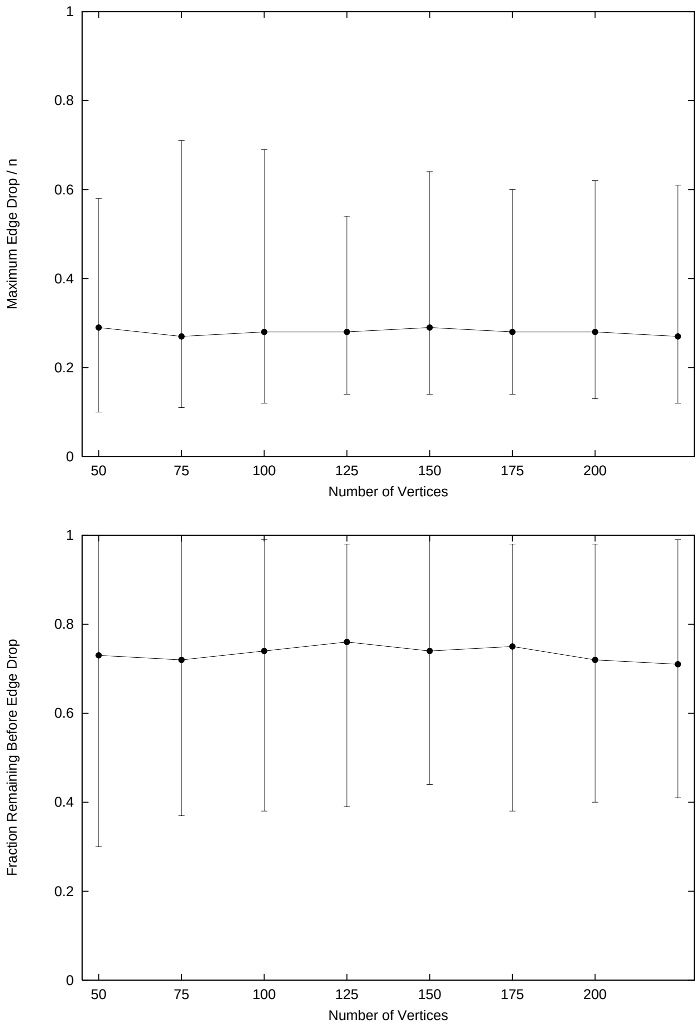


Fig. 5. (Top) The average maximum drop on a single edge. The error bars represent the minimum maximum drop and the maximum maximum drop over 100 samples. (Bottom) The minimum, average and maximum of the number of vertices in the graph at the top of the maximum drop. The average here too seems fairly consistent with the 72% observed before for $n = 200$. Observe that the drop on one edge is almost equal to the total drop up to that point on average; i.e. 100-72=28. However, the range can extend anywhere from under 40% to 100%.

the k-partite complete graph $\mathrm{G}_{\binom{n}{2}}(\pi)$.

For example, in 3-coloring, the sequence of edges 1-2,1-3,1-4,2-3,2-4,3-4,4-5,5-6, . . . , reduces to 1-2,1-3,1-4,2-3,2-4,4-5,5-6, . . . , with 3-4 omitted as it completes a 4-clique. Many different sequences reduce to the same scapegoat-free subsequence. In this example, the sequence 1-2,1-3,1-4,2-3,2-4,4-5,5-6,3-4, . . . reduces to the same subsequence.

We can use the frozen development to calculate exactly the probability of colorability over all sequences π' with the same scapegoat free subsequences as

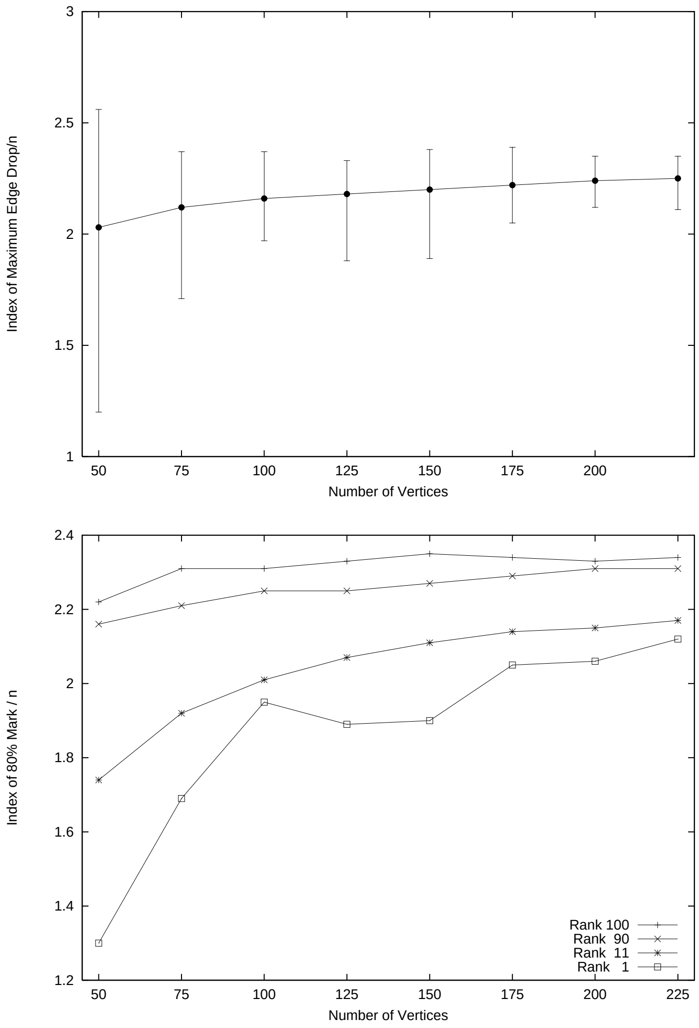


Fig. 6. (Top) The average index of the maximum drop edge. The error bars indicate minimum and maximum index over the entire set. (Bottom) The index (as a ratio to n) where the collapse first has fewer than 80% of its original vertices. The ranks are with respect to the 100 samples when sorted by this index. Thus, the rank 1 and rank 100 lines are the minimum and maximum index of each 100 sample set, while the other ranks bound the 80% of the samples in the middle, a simple minded attempt to remove the high variance extremes. Note the convergence apparent here, and also note that in no instance did the first drop below 80% exceed $2.35n$ edges.

π, i.e. $R(\pi') = R(\pi)$. Note that in this case $\mathbf{S}(\pi) = \mathbf{S}(\pi')$ also. We use the observation that this probability is also an unbiased estimator of the probability of colorability.[5] We explain these steps in detail before presenting empirical results.

Given a sequence π and its scapegoat-free subsequence $R(\pi)$, there is a family $\mathcal{R}(\pi)$ of sequences defined by $\mathcal{R}(\pi) = \{\pi' | R(\pi') = R(\pi)\}$. Given some index i,

[5] We thank David Wilson for pointing this out to us.

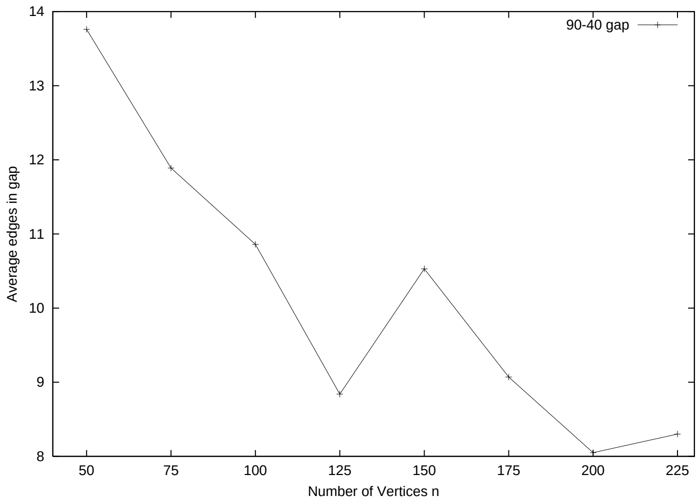


Fig. 7. The average over 100 samples of the number of edges in the gap between the first drop below 90% and the first drop below 40%. While noisy, the actual number of edges seems to be decreasing with n. For example, for 100 vertex graphs the addition of about 11 edges reduce the collapsed graph from 90 vertices to 40, while only 8 are necessary for $n = 200$ to reduce collapsed graphs from 180 to 80 vertices.

we can ask for a given $\pi' \in \mathcal{R}(\pi)$ whether or not the threshold index, $t(\pi')$, is greater than i or not; that is, if the first i edges in π' form a colorable graph. They will do so if and only if the first scapegoat in π' occurs at index $i+1$ or more. Given i, it is natural to ask what is the probability of colorability over $\mathcal{R}(\pi)$, i.e. the value of $\Pr\{t(\pi') > i | \pi' \in \mathcal{R}(\pi)\}$, assuming that each sequence $\pi' \in \mathcal{R}(\pi)$ is equally likely.

Our calculation of the entire frozen development of a sequence π allows us to exactly calculate the probability $P_{\pi,i} =_{def} \Pr\{t(\pi') > i | \pi' \in \mathcal{R}(\pi)\}$. We do so by induction, using the following equations.

$$\begin{aligned}
P_{\pi,0} &= 1 \\
P_{\pi,i} &= \Pr\{t(\pi') > i | \pi' \in \mathcal{R}(\pi)\} \\
&= P_{\pi,i-1} \Pr\{t(\pi') > i | \pi' \in \mathcal{R}(\pi) \ \& \ t(\pi') > i-1\} \\
&= P_{\pi,i-1} \Pr\{\pi'_i \notin \mathbf{S}(\pi) | \pi' \in \mathcal{R}(\pi) \ \& \ \pi'_1, \pi'_2 \ldots \pi'_{i-1} \notin \mathbf{S}(\pi)\}
\end{aligned}$$

The last step follows because the condition $t(\pi') > i-1$ implies that none of the first $i-1$ edges in π' can be scapegoats. This also implies that the first $i-1$ edges in π' are also the first $i-1$ elements in the scapegoat-free subsequence $R(\pi)$. The probability required in the final line can be calculated very simply if we have performed a full frozen development. The possible scapegoats are exactly the pairs frozen in the frozen development of π up to and including the index in π corresponding to the $i-1^{th}$ element of $R(\pi)$. Writing $P^C_{\pi,i}$ for the conditional probability $\Pr\{\pi'_i \notin \mathbf{S}(\pi) | \pi' \in \mathcal{R}(\pi) \ \& \ \pi'_1, \pi'_2 \ldots \pi'_{i-1} \notin \mathbf{S}(\pi)\}$, we have

$$
\begin{aligned}
P_{\pi,i} &= P_{\pi,i-1} P^C_{\pi,i} \\
P^C_{\pi,i} &= \frac{\binom{n}{2} - i + 1 - F_\pi(i-1)}{\binom{n}{2} - i + 1}
\end{aligned}
$$

$$\text{where } F_\pi(i) = |\{\pi_k : \pi_k \in \mathbf{S}(\pi), \iota(\pi_k) \leq j, \pi_j = R_i(\pi)\}|$$

In words, the definition of $F_\pi(i)$ is the number of frozen pairs in π up to the location in π where the i^{th} element of $R(\pi)$ appears. Note that the value of $F_\pi(i)$ is the same for all sequences $\pi \in \mathcal{R}(\pi)$.

We implemented a Perl script to calculate these numbers, given the frozen development over a permutation of pairs. To illustrate the nature of typical behavior, we show results in figure 8 of a single instance with $n = 225$. We first calculated the conditional probability $P^C_{\pi,i}$ We see a big jump in the conditional probability. This corresponds to the sudden jump in the number of frozen pairs. The probability of a *random* pair of nodes being consistent with any coloring is $\frac{2}{3}$. The conditional probability falls below this value because we are dealing with permutations of pairs, or in other words sampling without replacement from the set of edges. We were then able to calculate the unconditional probability of colorability $P_{\pi,i}$ for sequences with the same reduced subsequence as π, and this is shown in figure 8. Immediately after the conditional probability goes significantly away from 1, the absolute probability inevitably collapses. After that it decays exponentially, as can be seen in the log plot. This plot shows the advantage of our method: these probabilities are the *exact* probability of sequences with the same scapegoat-free subsequence being colorable with this number of edges. No such estimate was previously possible when the probability declined below about 10^{-9}, since samples of a billion would have been necessary, and even this would give only estimates instead of exact probabilities.

For random instances, the probability $P_{\pi,i}$ is of little direct interest, although it provides us with a valuable tool for studying non-random problems in the future. For random instances, the subject of this paper, the probability we are interested in is the global probability $P_i = \Pr\{G_{n,i} \text{ colorable}\}$. Fortunately, the probability $P_{\pi,i}$ is an unbiased estimator of P_i. That is, $E(P_{\pi,i}) = P_i$, where the expectation is over all sequences π.

To justify the claim of unbiased estimation, first note that

$$P_i = \Pr\{\text{The first } i \text{ edges in a random permutation are colorable}\}$$

This follows because over all permutations, each distinct graph G with i edges

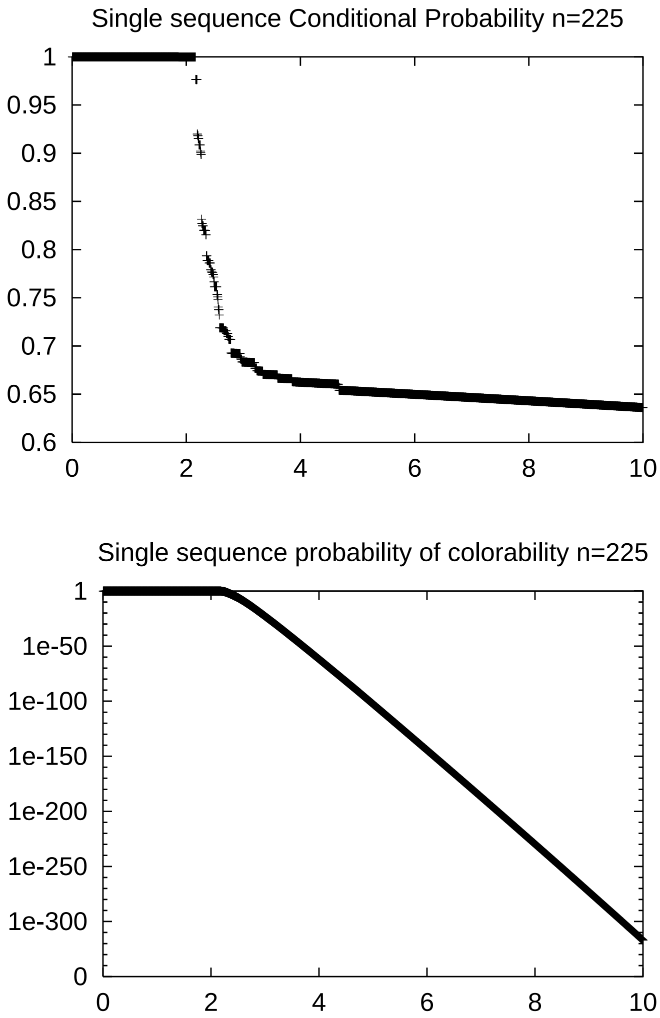


Fig. 8. (Top) The conditional probability $P^C_{\pi,i}$ for sequences with the same reduced subsequence as a particular π for $n = 225$, plotted against i/n. (Bottom) The overall probability of colorability $P_{\pi,i}$ for sequences with the same reduced subsequence as π, plotted on a log scale against i/n.

occurs exactly the same number of times, in particular exactly $i!(n-i)!$ times. We can easily prove that $P_{\pi,i}$ gives an unbiased estimator of this equivalent statement of P_i. The key to the proof is that the families $\mathcal{R}(\pi)$form a disjoint union of all sequences π. To formalize the simple proof, we introduce the indicator function $I_{\pi,i}$, which takes the value 1 if the first i edges in the sequence π are colorable, and 0 if not. We also extend the notation $P_{\pi,i}$ to $P_{\mathcal{R}(\pi),i}$ in the obvious way.

$$
\begin{aligned}
P_i &= \Pr\{\text{The first } i \text{ edges in a random permutation are colorable}\}\\
&= \frac{\sum_\pi I_{\pi,i}}{n!}\\
&= \frac{\sum_{\mathcal{R}(\pi)}(\sum_{\pi'\in\mathcal{R}(\pi)} I_{\pi',i})}{n!}\\
&= \frac{\sum_{\mathcal{R}(\pi)}(|\mathcal{R}(\pi)|P_{\mathcal{R}(\pi),i})}{n!}\\
&= \frac{\sum_\pi P_{\pi,i}}{n!}
\end{aligned}
$$

The combined result of this development is that we can introduce a new methodology for empirical estimation of probability of colorability. Instead of mere 'generate and test', in which we generate a sample of graphs from $G_{n,i}$ and test them for colorability, we instead generate random permutations and calculate the entire frozen development. From this, we can calculate for each sequence π and index i the value $P_{\pi,i}$ and use this as an estimate of P_i. Since this is an unbiased estimate, we can repeat this procedure for a sample of permutations and use the average to estimate the probability of colorability.

We can now reuse our data on frozen development to present estimates of probability of colorability from $n = 125$ up to $n = 225$, with a sample size of 100 permutations in each case. The results are shown in Figures 9 to 10.

Our experiments suggest that the technique is not very useful as a variance reduction technique where probability of colorability is significantly away from both 1 and 0. Where $P_i \approx 0.5$, the sample standard deviation of our estimates of P_i was about 0.4. We would expect a standard deviation of 0.5 using generate and test. Using a sample only about 50% bigger, we could obtain the same accuracy of estimation from generate and test, very much cheaper than by calculating full frozen development.

The value of our experiments is in giving estimates of probability in the highly colorable and the highly uncolorable region. This is particularly notable in the highly uncolorable region. We remind the reader that the estimates given are unbiased estimates of the true probabilities, even where values are as low as 10^{-300}. For the first time, we can use empirical data to picture the decay in probability of colorability. One caveat is necessary. Our estimates of probability are unbiased, but the true probability may be dominated by vanishingly rare instances. As a result, estimates obtained from comparatively small samples might be misleading and subject to very high variance. It might happen that of all possible permutations, almost all lead to an estimate $P_{\pi,i} \approx 10^{-300}$, but a fraction of one in 10^{-100} sequences gives an estimate of $P_{\pi,i} = 1$. In this hypothetical example, the true probability $P_i \approx 10^{-100}$, yet almost all small samples we take will suggest that $P_i \approx 10^{-300}$. We currently have no way of judging how accurate these estimates are. David Wilson suggested that the estimates might give the right *order of magnitude of the logarithm*, as in the artificial example just given [29].

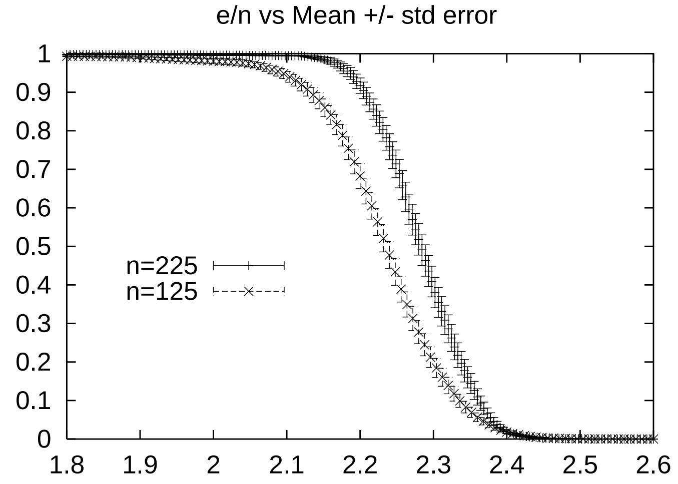


Fig. 9. A broad view of the phase transition in 3-COL for varying n, plotted according to the method introduced here. Error bars show +/- one standard error, i.e. sample standard deviation divided by the square root of sample size. In this case the sample was of 100 random permutations at each n. This figure shows little that would not be seen in a more conventional plot, except that the errorbars are slightly narrower than would be expected.

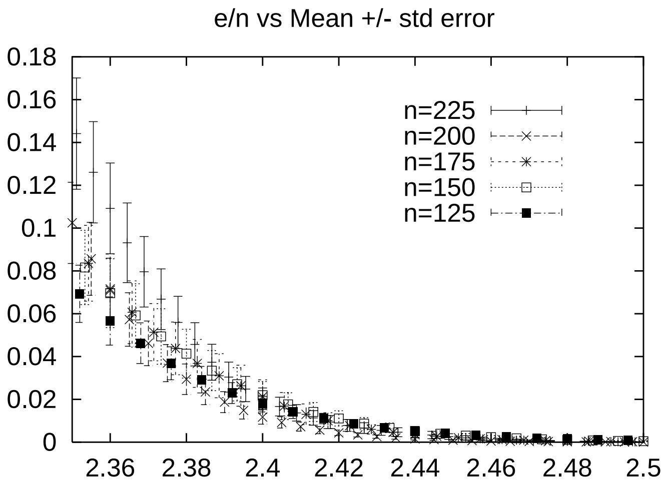
4 Hardness of Coloring Problems at the Threshold

We have seen a sudden jump in graph behavior. If this does connect with an asymptotic discontinuity in an order parameter, by analogy with SAT we should see that 3-colorability shows hard behavior at its threshold. Accordingly, we investigate this question in this section. Our evidence is strongly in favor of the belief that graph 3-coloring does indeed become hard at its threshold. This is also consistent with previous results from the literature [5,13]. In this section we present new empirical results to further support this claim.

In figure 11 we plot the growth in the average number of search nodes used to determine the colorability of the colorable (i.e. graphs with $t(\pi)-1$ edges) and uncolorable graphs at the threshold ($t(\pi)$). To obtain the threshold graphs for $n > 225$, we used our frozen development binary search to locate the threshold but did not compute the frozen sets. There are 100 instances at each n.

Both the median and average costs clearly exhibit exponential growth.

Notice that for small n the cost of the colorable graph is greater than the cost



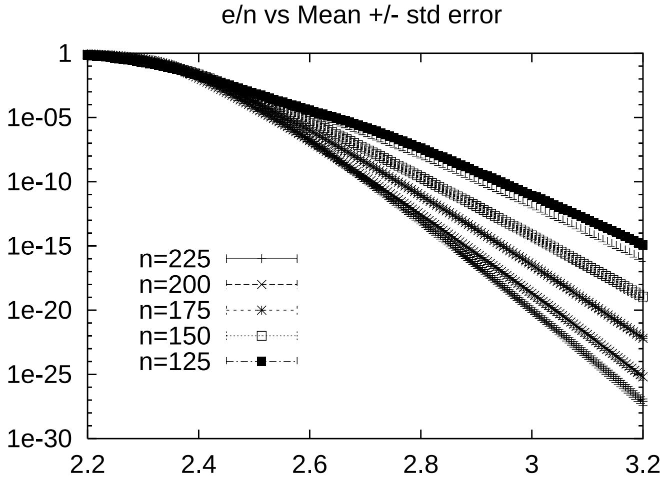


Fig. 10. (Top) A close up of the region where all n give very similar behavior. All error bars seem to cross at about $e/n \approx 2.4$. (Bottom)A close up of the highly insoluble region. Note the logarithmic scale, showing that beyond about $e/n = 2.8$ we are obtaining estimates of colorability entirely beyond the reach of conventional techniques.

of the uncolorable graphs with one more edge. This changes as n gets larger, with the uncolorable cost being more than twice as great at $n = 350$. The reason is that for the smaller values of n there is a high probability that the threshold graph contains a 4-clique, which is detected during initial pruning and so the search uses no backtrack nodes. As the analysis in Section 5 shows, these disappear with increasing n.

To study larger n, to get a picture of the distribution at various densities and to ensure that the problem is hard for algorithms other than our Smallk program, a series of experiments were run on graphs from the class G_{np}. In the first experiments random graphs were generated with the edge parameter

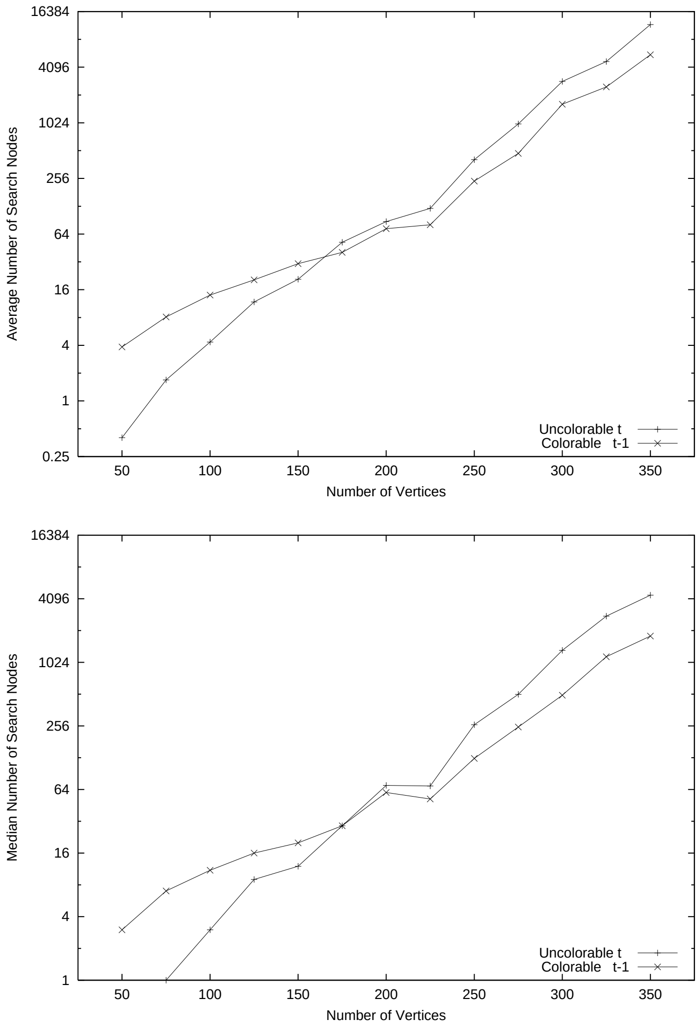


Fig. 11. The average (Top) and median (Bottom) number of search nodes used by Smallk for threshold graphs.

α ranging from 2.29 to 2.31, which is near the conjectured threshold. Recall that the expected number of edges is $\mathrm{E}[m] = \alpha n = p\binom{n}{2}$. Each graph was then tested using the Smallk program and the Ntab_back program. The Ntab_back is a SAT solver which implements the tableau method[7] with back jumping. It has been shown to be very efficient on short clauses, such as those generated by the conversion we describe next.

There are many ways to represent a coloring instance as a SAT instance. In preliminary tests we tried several of these and settled on the one in figure 12 (referred to as Version 2.5) as being the one that performed best with Ntab_back.

Version 2.5 considered here and used in the experiments begins to cross the line from a pure representation to one in which some solution information is carried in the representation. On considering the results for $k = 3, 4$ without

Variables $\forall v \in \mathbf{V}$ add $v_1, \ldots v_k$ to the set of variables $\mathbf{U}$. The meaning is that if v_i is true then vertex v gets color i.
Color Vertex $\forall v \in \mathbf{V}$ add $\langle v_1 \vee \ldots v_k \rangle$ to the set of clauses $\mathbf{C}$. The meaning is that if the clause is true then at least one of the colors is assigned to the vertex.
Edge Check $\forall \{u, v\} \in \mathbf{E}$, for each $i, 1 \leq i \leq k$ add $\langle \bar{u}_i \vee \bar{v}_i \rangle$ to $\mathbf{C}$. The meaning is that the same color cannot be assigned to both end points of an edge.
Only One Color $\forall v \in \mathbf{V}, \forall 1 \leq i < j \leq k$ add $\langle \bar{v_i} \vee \bar{v_j} \rangle$ to $\mathbf{C}$. This means at most one of the colors can be assigned to v.
Symmetry Breaking For exactly one edge $\{u, v\} \in \mathbf{E}$ add the two clauses $\langle u_1 \rangle$ and $\langle v_2 \rangle$ to $\mathbf{C}$.

Variables: nk
Clauses: 2 1-clauses; $mk + n\binom{k}{2}$ 2-clauses; n k-clauses

Fig. 12. Version 2.5 used to represent a k-coloring instance $\langle \mathrm{G} = (\mathbf{V}, \mathbf{E}), k \rangle$ as an instance $(\mathbf{U}, \mathbf{C})$ of SAT. Here $n = |\mathbf{V}|$, $m = |\mathbf{E}|$.

the symmetry breaking, it became evident that one of the difficulties faced by the SAT search engine especially on unsolvable instances, was that it had no way of preventing searches that basically relabeled the color classes.

In Version 2.5 symmetry breaking information is presented to SAT. Basically an edge is detected in G and the two vertices are forced to distinct colors 1 and 2. This is a trivial addition to a conversion program since edges must be checked anyway. However, it does open a can of worms if our goal is to check the relative efficiency of two approaches. If we are allowed to add arbitrarily to the converter, then with sufficient effort we could simply encode the solution, which the SAT program would merely have to output. Of course we could add the time for conversion to the time for the SAT routine, but this opens up other questions of efficiency of the conversion.

Similarly, the "**Only One Color**" clauses are not strictly necessary. But again preliminary tests indicated these 2-clauses occasionally helped on the harder instances for small values of k, and seldom caused Ntab_back to require more time. A number of other conversions were also tested, but none were competitive with this for $k = 3, 4$.

Using this conversion on the instances of 3-coloring considered here, usually the two programs were within a factor of 8 or less in run time, with both "winning" equally often. Conversion times were not considered, being only a tiny fraction of the search time. Rarely, one or the other program would take considerably longer than the other, up to a factor of 50 or more. This typically occurred on instances that were solvable, where one program would get lucky and solve the instance very quickly. For $k > 4$, Smallk was almost invariably faster, frequently by orders of magnitude.

For this section the minimum time from the two programs was extracted. [6] This minimum time is used in as the indicator for the growth rate of the solving cost. [7] In figure 13, we present evidence that the cost is growing exponentially, even at the 25% rank. Our data suggests search cost grows as $\approx 2^{n/25}$.

There is a large difference between the typical cost of solvable and unsolvable instances. In figure 14 the instances are split into those that were colorable and those that were not. For $n \leq 500$ a time limit of one hour was placed on each program. This time limit was exceeded by both programs in three instances: one instance at $n = 450$ and density $\alpha = 2.30$, and two instances at $n = 500$ and $\alpha = 2.31$. These are included as unsatisfiable instances in figure 14. For $n = 550$ the time limit was removed, but there are only 25 instances at each data point. Both satisfiable and unsatisfiable instances appear to be increasing in difficulty at an exponential rate, with an almost constant ratio in median times between the two cases.

To obtain a better picture of where the hard instances are, sampling was done for $n = 500$ with α ranging from $\alpha = 2.19$ to $\alpha = 2.64$. The profile for three ranks is plotted in figure 15. We also indicate the number of unsatisfiable instances at each sample point, demonstrating that the plotted region encompasses the entire transition from colorable to uncolorable for these instances.

These figures support the contention that there is a broad band of hard instances at $n = 500$. In fact, the frequent hard instances occur well beyond the 2.31 threshold where 50% of the instances become unsolvable to $\alpha = 2.4$ and beyond.

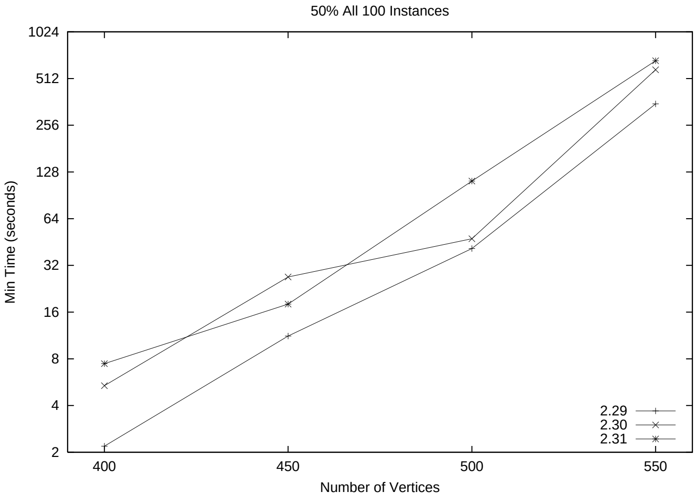
As further evidence a series of tests were run at $n = 700$. Extrapolating from figure 13 we would expect the median cost at $\alpha = 2.31$ to be in the region 10-15 hours for these sample points, making extensive unbounded tests infeasible. Instead, we opted for a time limit of one hour, using only the Smallk program. [8] The results show that indeed the median cost is greater than 1 hour in many

---

[6] In two cases Ntab_back incorrectly asserted that there was no solution. In these instances the time from Smallk was used, but in others we continue to use the Ntab_back time on the assumption that a debugged Ntab_back would probably give similar results to the version we used.

[7] For an accurate estimate of resource usage, this time should be multiplied by 2.

[8] For the harder instances it is rare for the two programs to be out by more than a factor of 4 to 8, so we do not expect significantly different results if Ntab_back were also used.



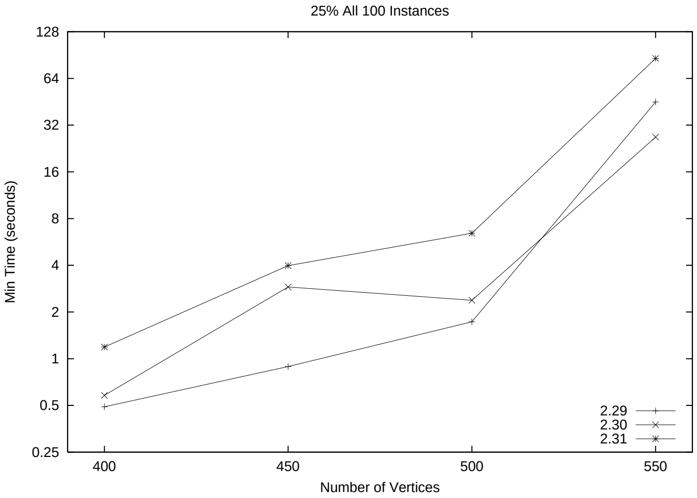


Fig. 13. The growth rate in the cost of coloring or proving uncolorable. There are 100 instances at each of 400, 450 and 500, but only 25 instances at 550. Three edge densities are reported, from 2.29 to 2.31. (Top) The 50% rank. (Bottom) The 25% rank.

cases. Given that the median cost at $n = 500$ does not exceed 121 seconds for any α, growth to a median in excess of 3600 seconds at $n = 700$ reflects a growth rate of at least n^{10} over this range, sufficient to convince us that the instances are typically hard. We thus believe that the presentation in figure 16 is a reasonable representation of the difficulty region at $n = 700$. This evidence is consistent with the conjecture that the hard region is in fact growing in width (as a ratio to n) as n increases, with 50% or more of the instances being hard from α less than 2.3 to 2.5 or more.

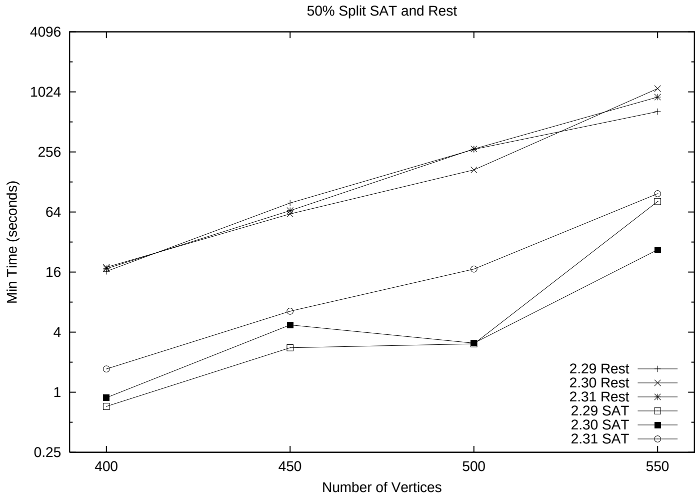


Fig. 14. Splitting the instances into those that are satisfiable and those that were not at the 50% rank indicates that both are growing exponentially, although the satisfiable instances are more than an order of magnitude easier. “Rest” indicates that in three cases both programs exceeded the one hour time limit, but were assumed to be unsatisfiable.

5 Critical Graphs and Thresholds

The most fundamental question that must be addressed in understanding the link between the nature of the order parameter and the difficulty algorithms have in solving instances is the nature of the constructs that cause the algorithms difficulty. For graph coloring, almost all search algorithms proceed by restricting the set of colorings in some way, either by assigning colors to selected vertices, or by restricting the available colors for vertices, or by modifying the graph in some way that reflects a restricted set of colorings such as merging two independent vertices or adding an edge. Some algorithms (Smallk) use a combination of these techniques.

When a graph is uncolorable, all such methods must eventually fail. If a graph is colorable, then the increasing restrictions can create a situation in which there is no solution. In either case, the graph (with possibly some coloring restrictions) will contain a subgraph which is sufficient to show uncolorability. The efficiency of these algorithms depends on finding and verifying (at least implicitly) at least one such uncolorable substructure.

A graph $\mathrm{G} = (\mathbf{V}, \mathbf{E})$ is $k+1$-*edge-critical* if it is not k-colorable, but every subgraph on a proper subset $\mathbf{E}' \subset \mathbf{E}$ is k-colorable. We use the term critical to mean edge-critical. Clearly, such a G is $k+1$-colorable. It is apparent from the preceding paragraphs that the size, number and structure of critical subgraphs of an uncolorable graph are of paramount importance to the efficiency of most search algorithms. Studying critical substructures can help us understand

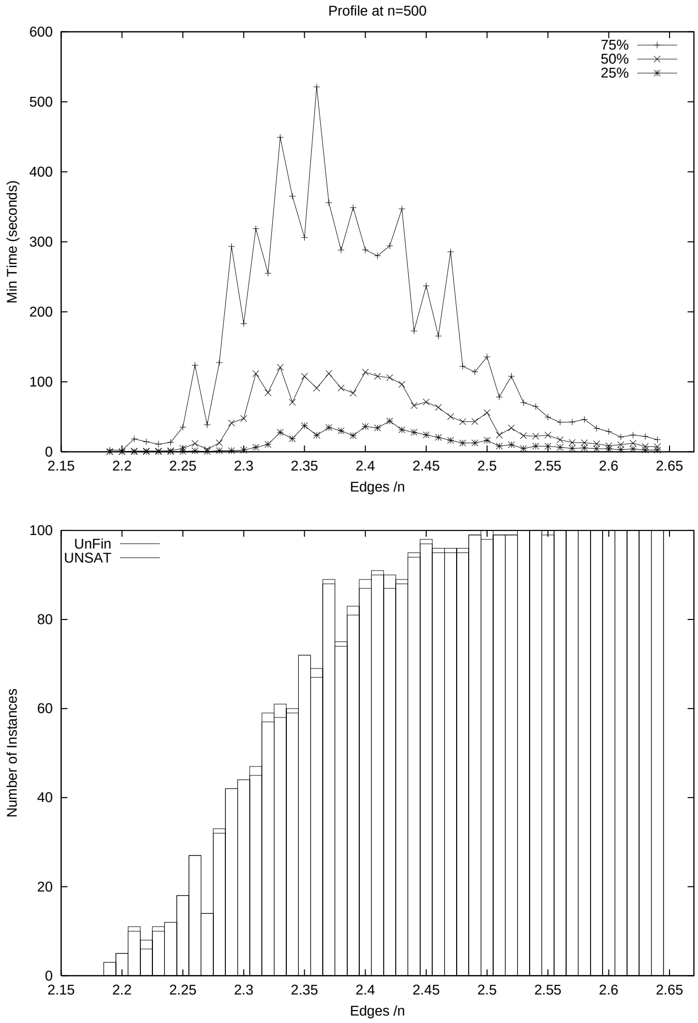


Fig. 15. (Top) The cost of the 25%, 50% and 75% ranks at $n = 500$ for $\alpha = 2.19 \ldots 2.64$. Samples were taken at every 0.01. There are 100 instances at each sample point. (Bottom) The number of instances that were not satisfiable. The little tops indicate the number of instances on which both Smallk and Ntab_back exceeded the one hour time limit. It is likely that most of these were unsatisfiable as well.

hard instances [16,9,1]. We observe that an instance cannot be hard if it is unsolvable for reasons detected by an algorithm in polynomial time: in the case of coloring this occurs when instances contain a small critical subgraph. For example, non 3-colorable graphs are easy to detect if they contain a 4-clique.

It is well known (see e.g. Palmer [23] chapter 3) that the threshold probability for 4-cliques for graphs drawn from the $\mathrm{G}_{n,p}$ class is $1/n^{2/3}$. This means $O(n^{4/3})$ is an upper boundary on the number of edges in random graphs that may be hard, since programs such as Smallk easily detect 4-cliques.

In appendix I we show that 4-critical subgraphs are of order $\Omega(n^{\varepsilon})$ for $p \in$

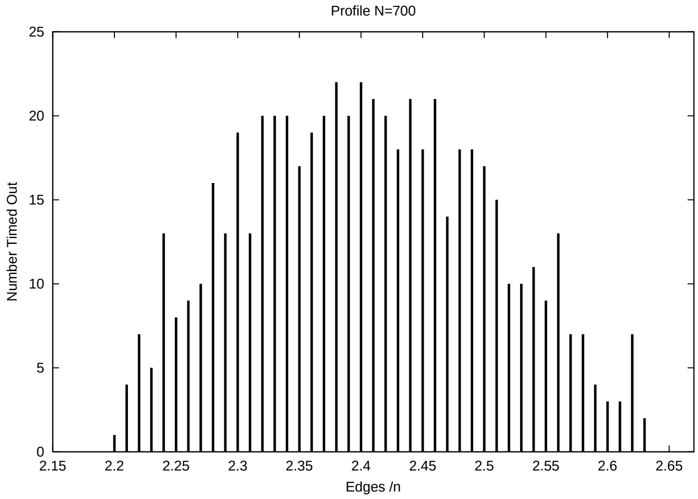


Fig. 16. The number of instances that were not completed in one hour, $n = 700$, 25 instances at each sample.

$O(1/n^{2/3+\epsilon})$, where $0 < \varepsilon < \epsilon/2.5$. Note that this probability corresponds to an expected number of edges $\mathrm{E}[m] \in O(n^{4/3-\epsilon})$.

This result suggests that the hard region could extend to graphs with density arbitrarily close to $O(n^{4/3-\epsilon})$. The reason is that it would appear to be very hard to detect and verify critical subgraphs of order n^{ϵ} in graphs of order n. In fact, just to detect cliques of this order is already NP-complete, and since large critical graphs are not cliques one would expect the non-colorability problem to be even more difficult. Of course, for small n such as we use in our experiments 4-cliques will usually occur for graphs with quite a bit less than $n^{4/3}$ edges, but we still expect, and observe, that the hard region extends well beyond the colorability threshold.

Although large critical subgraphs are necessary for hard non-colorable instances, being large is not sufficient. Starting with 4-cliques, arbitrarily large 4-critical graphs can be constructed using only the *Hajós join construction* [11] [9] which, even when embedded as subgraphs in larger graphs, could be recognized immediately using techniques similar to those in Smallk as not having a 3-coloring. The reason is that in these graphs there are always at least two near-4-cliques (n4c's; 4-cliques with one edge removed). In n4c's the two independent vertices are frozen, that is they must be the same color. These lead to chains of collapse that show immediately that the graph is not 3-colorable.

Thus, the structure of the critical graphs must be such that they are not easily

[9] Given two non-colorable graphs G_1 with edge $\{x, y\}$ and G_2 with edge $\{v, w\}$, the join construction creates a new graph by deleting the two edges, merging the two vertices x and v, and adding the edge $\{y, w\}$.

recognized. It is known that the threshold for n4c's is $1/n^{4/5}$ [23, chapter 3]. Thus, for random graphs with $o(n^{6/5})$ edges these will disappear asymptotically. By considering the expectation formula, we find that for $p = (n(n-1)(n-2)(n-3)/4)^{-1/5}$ the expected number of n4c's is approximately one. The ratio of the expected number of edges to vertices at $n = 700$ for this p is 2.45, which is a little greater than the $\alpha = 2.3$ threshold for three coloring. This range compares well to the top of the hump in figure 16. However, hard instances occur well above even this boundary. Apparently the presence of a few n4c's is not enough to make the coloring task easy.

What may not be apparent from the analysis is that as we increase the number of nodes, when the 4-cliques disappear we see a sudden jump in the size of 4-critical graphs to ones with $\Omega(n)$ edges. As evidence we present data from three measurements on threshold graphs.

First we use a simple program to compute the number of graphs containing a 4-clique out of each set of 100 instances. We note that in all cases the threshold graph has either zero or one 4-clique, which is to be expected since the addition of a single edge is very unlikely to create two or more at once.

Second for each instance, we generate a random permutation of the edges of the threshold graph, then delete the edges in order, replacing each edge that causes the graph to become colorable. The graph that remains is necessarily a $3+1$-critical subgraph of the threshold graph. This may be larger or smaller than the average size over all critical graphs, but there is a straight forward argument that smaller ones will be the most probable. Of note, this process found the 4-clique in every graph that contained one. It would be interesting to determine the minimum critical subgraph in each instance. Unfortunately this will likely prove difficult, possibly almost as hard as listing all critical graphs. Further investigation is needed on how to compute all critical graphs [26].

Third, we compute the critical set of each threshold graph. The *critical set* of edges in G is the set of edges that occur in every critical subgraph of the graph. Thus, these act as a lower bound on the size of the smallest critical subgraph, although the smallest critical subgraph can be arbitrarily larger than the critical set of the graph. Note that an edge is in the critical set iff its removal from G makes G colorable. Thus, to compute the set, we simply test each edge to see if its removal makes the graph colorable.

In table 2 we show a breakdown of critical subgraphs of the threshold non-3-colorable graphs. We have separated out very small critical graphs, that is 4-cliques and other small graphs with up to 12 edges, and other critical graphs. Two features of table 2 are particularly important. First, we see a reduction in the number of small critical graphs, from more than 10% of cases at $n = 50$

Critical Subgraphs of Threshold							
Small Cases			Remaining Cases				
n	#4-cliques	#≤ 12	#	Min Size	Max Size	Mean	Mean/n
50	11	2	87	15	80	48.09	0.96
75	2	0	98	16	151	79.97	1.07
100	0	0	100	45	174	114.73	1.15
125	3	0	97	78	211	162.03	1.30
150	3	0	97	36	276	202.90	1.35
175	2	0	98	134	332	259.91	1.49
200	1	0	99	227	384	310.02	1.55

Table 2
Size distributions of the critical subgraphs of threshold graphs, $k = 3$. 100 samples were taken at each n. For $n = 225$ there were no 4-cliques, but we did not do the subgraph computation for reasons of time.

to only a few percent. At $n = 200$, there is only one occurrence of a 4-clique, and the next smallest critical graph we observed contains 227 edges. This data seems consistent with the hypothesis that there will be a sudden jump in the size of the critical subgraphs. Second, the growth in size of the larger critical graphs is striking. Indeed, these graphs are growing *faster* than linearly with n. That is not possible asymptotically, since the threshold occurs at $O(n)$. We take this as evidence that critical graph size is $O(n)$, and we assume that the apparent superlinear growth is towards some fixed ratio of critical graph size to n.

By computing critical sets, we have a lower bound on the size of the smallest critical graph. All edges in the critical set must be examined by any algorithm confirming uncolorability. In table 3 we report the number of edges in critical sets at the threshold, and in figure 17 we show the distribution of sizes of critical sets for $n = 225$. The minimum may obtain its minimum possible value of 1, since the scapegoat edge is always critical. Despite this, we see a mean size of critical set of $\approx 0.35n$. This suggests that the critical set is growing linearly, as is the critical graph size.

The result that critical graphs are $O(n)$ at the threshold and that they grow very quickly to this size once the graphs are large enough to prevent 4-cliques means that more intelligent algorithms cannot be expected to perform significantly better, unless some remarkable new method for proving uncolorability with structures other than subgraphs is discovered. Thus phase transition instances can be expected to be hard for all algorithms. We have reported experiments elsewhere with 4-coloring and triangle-free random graphs for 3-coloring [8]. In both cases almost all critical graphs were hard, and there was

Size of Critical Sets					
N	Avg	Std	Ratio	Min	Max
50	16.800	11.615	0.336	1	49
75	26.960	18.675	0.359	1	83
100	31.850	21.119	0.319	1	85
125	41.500	29.915	0.332	3	160
150	51.180	34.633	0.341	5	140
175	70.400	49.407	0.402	1	195
200	69.730	52.727	0.349	2	205
225	79.910	57.830	0.355	1	256

Table 3
The critical sets, that is the number of edges that must occur in every critical subgraph of the threshold non-3-colorable graphs. Notice that there is a distinction between scapegoats and edges in the critical set of the threshold graph. Given the sequence π used to generate the graph, the scapegoat is uniquely the last edge added to form the uncolorable threshold graph. This edge must be in the critical set, because its removal makes the graph colorable. There may be many other edges in the critical set.

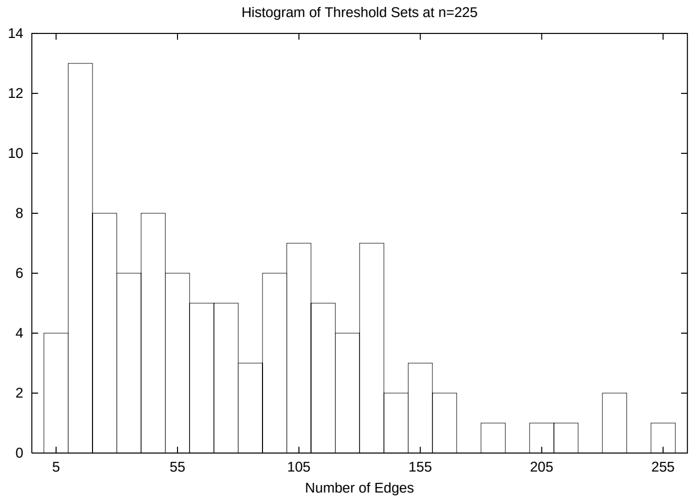


Fig. 17. Distribution of the number of edges in the critical set at $n = 225$. These edges are those in the threshold graph whose removal makes the graph colorable. There is always at least one such edge, i.e. the scapegoat added as the last edge in the threshold graph. Exactly one graph in this set had an edge set of only one edge.

very little distance between the first frozen and threshold points. We conjecture that similar results will apply to other NP-complete problems in which phase transitions provide hard instances. It would be especially interesting to study problems such as the TSP, in which the analogue of critical graphs are not so obvious.

6 Backbones versus Frozen Sets and Future Research

The fact that there are large ($O(n)$) critical sets may also have implications for empirical attempts to model statistical mechanical analysis, for example by measuring backbones. We discuss this briefly, and suggest some avenues of further research.

We have defined a threshold with respect to the frozen development process as the edge which when added causes the graph to become non-k-colorable. Let us refer to this as the t_0 threshold. The subscript 0 refers to the fact that we will 3-color the graph with no edge violations [10].

We suggest that we can take the frozen development measure for a sequence of thresholds, $t_0, t_1, \ldots$. We say that a graph is (k, v)-colorable if it can be k-colored by violating at most v edges.

We say that a pair of vertices (x, y) is (k, v)-frozen under (k, v)-colorings if

(1) G *is (k, v)-colorable,*
and
(2) for every (k, v)-coloring c of G *we have $c(x) = c(y)$.*

Note that if we set $v = 0$ then this is exactly the same frozen we have used previously. Also, if a pair is (k, v)-frozen in G_m, then it is (k, v)-frozen for every (k, v)-colorable graph $\mathrm{G}_{m'}$ in the sequence where $m' > m$.

Now we are ready to define the multiple thresholds. We define the (k, v)-threshold as

$$t_v = t_v(\pi) = \min\{m : \pi_m \text{ is } (k, v)\text{-frozen at } m - 1\} \tag{1}$$

In an analogous manner we can also define (k, v)-scapegoats, the (k, v)-spine and so on to complete a (k, v) frozen development process. Although in principle we can now use our frozen development process, the cost of doing so will be very high.

It is known that backbones are not monotonic, that is, the size of the backbone may shrink as edges are added to a graph[4]. By considering the size of the critical sets at the threshold, we can get some indication of the magnitude of

[10] An edge is violated by a coloring c if both endpoints receive the same color.

the possible drop. Note that below the index t_0 of the threshold graph, the backbone is identical to our frozen set.

What happens on the addition of the threshold edge? When the t_0 threshold edge is added, then every $(3,1)$-coloring must violate exactly one of the edges in the $(3,0)$-critical set. The backbone measure now switches from being defined by the set of $(3,0)$ colorings to being defined by the set of $(3,1)$ colorings.

There are two things to notice. First every $(3,1)$ coloring of the t_0 graph that violates the threshold edge is in fact a $(3,0)$ coloring of the $t_0 - 1$ graph. Secondly, for every other edge in the critical set of the t_0 graph there will be a set of distinct $(3,1)$ colorings. These sets are all distinct, since for each edge in the critical set that pair of vertices receive the same color only in those colorings that violate that edge. The data in Table 3 show that there are approximately $0.35n$ edges on average in the critical set.

We can expect then that the number of $(3,1)$ colorings at t_0 should be about $0.35n$ times as many as the number of $(3,0)$ colorings at $t_0 - 1$. This statement makes the assumption that there is nothing particularly special about the threshold edge, since any other edge of the critical set could have occurred at the threshold under an equal number of permutations. This strongly suggests that the number of pairs in a backbone measure could be sharply reduced at t_0 because it is likely that many frozen pairs will be unfrozen when the number of distinct colorings goes up by a factor of $0.35n$.

Now consider a random sampling of graphs near the threshold, that is sampled below the t_1 threshold. Graphs which happen to be below the sudden jump will not exhibit a large backbone. Graphs which occur shortly after the t_0 threshold may not exhibit a large backbone, as argued above. The only graphs we can have confidence of exhibiting a large backbone are those that have exhibited the large jump, but are still 3-colorable. Considering the evidence of the preceding sections, these are a very tiny fraction of all m-edge graphs for any fixed m, and would also be a very tiny fraction of all graphs generated using the G_{np} model for any p.

Now we become somewhat more speculative. We suspect that a similar effect holds at each t_i. If so, then an experimental approach that randomly generates graphs and measures the backbone with respect to minimum violation colorings might give little or no evidence of the sudden jump in frozen sets our methods are showing.

The extent of these effects, and the nature of the (k,v) thresholds and frozen sets could be the topic of future research.

7 Conclusions

Our contributions in this paper are in two areas: the frozen development of graph coloring as we add edges to graphs; and the reasons why graphs found near the colorability threshold are hard to color.

We have described the notion of 'full frozen development' of graphs. This gives rise to a definition of the spine of a graph, analogous to the backbone or spine of a satisfiability problem. We have shown that the full frozen development can be calculated in $O(n^2 \log n)$ calls to a graph coloring program, and reported the practical algorithm we use to calculate full development up to $n = 225$. We also showed that this can be used to calculate unbiased estimates of the probability of colorability in regions even where this probability is $\approx 10^{-300}$, and we reported empirical estimates of probability using this method.

We have reported a number of novel empirical results on the development of the spine. We showed empirically that the spine of a graph shows a dramatic jump, usually just before the threshold in colorability. Since this measure is based on elements (pairs) which are quadratic in the number of vertices, we also converted this measure to one of counting the number of equivalence classes forced by the set of three colorings. This results in what we call a 'collapsed' graph, which shows a sharp drop in size corresponding to the jump in frozen pairs. The collapse is always as dramatic as the freezing. Every instance we studied showed a collapse of at least 10% in size when a single edge is added, and the average was 28%.

In terms of the difficulty of coloring, we reported empirical results demonstrating that there is a widening range over which the best programs available, including a conversion to SAT, show exponentially (in n) increasing median time. This median growth rate is $\approx 2^{n/25}$ over the range examined. We then analyzed the nature of critical graphs at thresholds, theoretically and empirically. Unless remarkable new methods are discovered, large critical graphs correspond to hard instances, because algorithms have to investigate a large number of edges to prove uncolorability. We observed that at small n, very small critical graphs such as 4-cliques do occur, but that when these disappear there is a sudden jump to critical graphs of size $O(n)$. This result agrees with prior conjectures, and is supported by theoretical results on hardness of coloring[17].

We are aware that the spine we have introduced may not be the order parameter for 3-COL. Our definition is similar to the spine in SAT defined by Borgs et al [4], although not exactly analogous. A minimum edge violation coloring approximation, would be closer to the backbone measure of Monasson et al [22]. If we used a backbone-like measure, this sharp (discontinuous) change might be reduced or eliminated. The empirical evidence supporting this is based on the critical sets, that is the set of edges such that the removal of any one of them would make the uncolorable graph colorable. At the threshold, this set is on average large, possibly $0.35n$ or larger. This means that allowing one edge to be violated might cause the t_0 threshold to exhibit few or no frozen pairs. To verify this conjecture we will probably need to make our programs more efficient, as the number of colorings needed could be significant.

We hope that these results will contribute to the analysis using statistical mechanics of phase transitions in graph coloring. In particular, the occurrence of a jump in freezing and a collapse in graph size is highly suggestive of a discontinuity of an the order parameter. In satisfiability, such a discontinuity has been correlated with the hardness of 3-SAT instances. Since we have reported results suggesting both a discontinuity and the hardness of 3-COL instances, we hope that future investigations will uncover a link like that found in SAT.

References

[1] R.J. Bayardo Jr. and R.C. Schrag. Using CSP look-back techniques to solve exceptionally hard SAT instances. In *Principles and Practice of Constraint Programming – CP96*, pages 46–60. Springer, 1996.

[2] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3446^n)$: a no-MIS algorithm. In *36th IEEE Symposium on Foundations of Computer Science*, pages 444–452, 1995.

[3] B. Bollobás. *Modern Graph Theory.* Graduate Texts in Mathematics 184, Springer, 1998.

[4] B. Bollobás, C. Borgs, J.T. Chayes, J.H. Kim, and D.B. Wilson. The scaling window of the 2-SAT transition, 1999. Paper in the xxx.lanl.gov e-Print archive. Available from http://xxx.lanl.gov/abs/math.CO/9909031.

[5] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, pages 331–337, 1991.

[6] V. Chvatal and E. Szemeredi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, 1988.

[7] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–57, 1996. Code at www.cirl.uoregon.edu/crawford.

[8] J. Culberson and I.P. Gent. Well out of reach: why hard problems are hard. Technical Report APES-13-1999, APES Research Group, 1999. Available from http://apes.cs.strath.ac.uk/apesreports.html.

[9] I.P. Gent and T. Walsh. The satisfiability constraint gap. *Artificial Intelligence*, 81:59–80, 1996.

[10] F. Glover, M. Parker, and J. Ryan. Coloring by TABU branch and bound. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 285–307. American Mathematical Society, 1996.

[11] Denis Hanson, Gordon C. Robinson, and Bjarne Toft. Remarks on the graph colour theorem of Hajós. *Congressus Numerantium*, 55:69–76, 1986.

[12] Brian Hayes. Can't get no satisfaction. *American Scientist*, 85:108–112, March 1997. http://www.amsci.org/amsci/issues/Comsci97/compsci9703.html.

[13] Tad Hogg and Colin P. Williams. The hardest constraint problems: A double phase transition. *Artificial Intelligence*, 69:359–377, 1994.

[14] T. R. Jensen and B. Toft. *Graph Coloring Problems*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1995.

[15] A.D. Korshunov. The main properties of random graphs with a large number of vertices and edges. *Russian Math. Surveys*, pages 121–198, 1985.

[16] D.L. Mammen and T. Hogg. A new look at the easy-hard-easy pattern of combinatorial search difficulty. *Journal of Artificial Intelligence Research*, 7:47–66, 1997.

[17] Colin McDiarmid. Determining the chromatic number of a graph. *SIAM Journal on Computing*, 8(1):1–14, February 1979.

[18] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of AAAI-92*, pages 459–465, 1992.

[19] R. Monasson and R. Zecchina. Statistical mechanics of the random k-sat model. *Phys. Rev. E*, 56, 1997.

[20] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Phase transition and search cost in the 2+p-sat problem. In T. Toffoli, M. Biafore, and J. Leao, editors, *Proceedings of PhysComp 96*, Boston, 1996. http://www.lpt.ens.fr/ monasson/ http://www.ictp.trieste.it/ zecchina.

[21] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400:133–137, 1998. http://www.lpt.ens.fr/ monasson/ http://www.ictp.trieste.it/ z ecchina.

[22] R. Monasson, R. Zecchina, B. Selman, and L. Troyansky. 2+P-SAT: Relation of typical-case complexity to the nature of the phase transition:, 1999.

[23] E.M. Palmer. *Graphical Evolution: An introduction to the Theory of Random Graphs*. John Wiley & Sons, 1985.

[24] A. Parkes. Clustering at the phase transition. In *Proc. AAAI-97*, pages 340–345, 1997.

[25] P. Prosser. An empirical study of phase transitions in binary constraint satisfaction problems. *Artificial Intelligence*, 81:127–154, 1996.

[26] I. Ross. Finding all muses, 1997. Personal communication, isla@cs.strath.ac.uk.

[27] B.M. Smith and M.E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.

[28] B. Vandegriend and J. Culberson. The G_{nm} phase transition is not hard for the Hamiltonian cycle problem. *Journal of Artificial Intelligence Research*, 9:219–245, 1998. www.jair.org/contents/v9.html.

[29] D.B. Wilson. Personal communication, Sep. 1999.

[30] R. Zecchina. personal communication.

Appendix I: Analysis of the Growth of 4-Critical Subgraphs

Our goal is to show that the non-k-colorable graphs generated under the $\mathrm{G}_{n,p}$ with p varying over some range, have large $k+1$-critical subgraphs.

To show that there are no small critical subgraphs we use the standard technique of bounding the expected number of at most c-vertex $k+1$-critical subgraphs, for $k \geq 3$, and showing this expected number goes to zero.

Under the $\mathrm{G}_{n,p}$ random graph model we let $p = \frac{2}{(n-1)^y}$. The expected number of edges is then $\mathrm{E}[m] = n(n-1)^{1-y} \in O(n^{2-y})$. We will show that $y \geq 2/k + \epsilon$, for $\epsilon > 0$, will suffice to imply that the smallest $k+1$-critical graph will have $c \geq (n-1)^{\varepsilon}$ vertices (whp).

First note that the number of edges h in a critical graph must be at least $h \geq \frac{k}{2}c$, since the minimum degree of a critical graph is at least k. Now, the expected number of c vertex critical subgraphs must be bounded by the expected number of c vertex graphs with minimum degree k whence

$$\mathrm{E}\left[\#\mathrm{Crit}\right] < \sum_{j=k+1}^{c} \sum_{h=\frac{k}{2}j}^{\binom{j}{2}} \binom{n}{j} p^h (1-p)^{\binom{j}{2}-h} \binom{\binom{j}{2}}{h} \tag{2}$$

$$\leq \sum_{j=k+1}^{c} \sum_{h=\frac{k}{2}j}^{\binom{j}{2}} \frac{n^j}{j!} p^h \frac{(j^2/2)^h}{h!} \tag{3}$$

$$\leq \sum_{j=k+1}^{c} \sum_{h=\frac{k}{2}j}^{\binom{j}{2}} \frac{(j^2)^h}{(n-1)^{yh-j}} \tag{4}$$

$$\leq \sum_{j=k+1}^{c} \sum_{h=\frac{k}{2}j}^{\binom{j}{2}} \left[\frac{j^2}{(n-1)^{y-j/h}}\right]^h \tag{5}$$

$$\leq \sum_{j=k+1}^{c} \sum_{h=\frac{k}{2}j}^{\binom{j}{2}} \left[\frac{j^2}{(n-1)^{y-2/k}}\right]^{kj/2} \tag{6}$$

$$\leq \sum_{j=k+1}^{c} j^2 \left[\frac{j^2}{(n-1)^{y-2/k}}\right]^{kj/2} \tag{7}$$

Assuming that c is chosen so that the term in brackets is less than one for sufficiently large n we have

$$\leq c^3 \left[\frac{c^2}{(n-1)^{y-2/k}}\right]^{k(k+1)/2} \tag{8}$$

At this point we can easily see that if $y > 2/k$ then for any fixed constant c this goes to zero.

To get a better bound on c, we take the logarithm

$$\ln \mathrm{E}\left[\#\mathrm{Crit}\right] < 3 \ln c + \frac{k(k+1)}{2} \left[2 \ln c - (y - 2/k) \ln(n-1)\right] \tag{9}$$

Letting $c = (n-1)^{\varepsilon}$

$$\leq 3\varepsilon \ln(n-1) + k(k+1)\varepsilon ln(n-1) - (k+1)(yk-2)/2 \ln(n-1) \tag{10}$$

This goes to negative infinity if

$$\varepsilon(3 + k(k+1)) < (k+1)(yk-2)/2 \tag{11}$$

$$\varepsilon < \frac{(k+1)(yk-2)}{6 + 2k(k+1)} \tag{12}$$

For $k = 3$, $y = 2/3 + \epsilon$ we find for any positive $\varepsilon < \epsilon/2.5$ there will be no 4-critical graphs with $c = (n-1)^{\varepsilon}$ or fewer vertices.

Appendix II: Smallk

In this appendix we give a brief description of the Smallk coloring program, which is used throughout the frozen development process whenever it is necessary to color any graph. For reasons of space, this description is necessarily incomplete.

We define a list coloring problem[2,14] as a graph G $= (\mathbf{V}, \mathbf{E})$ where each vertex has a set of available colors, $\mathbf{A}(v)$, also called its availset. For our algorithm, we first formulate a k-coloring problem as a list coloring, initializing the availsets $\mathbf{A}(v) = \{1, \ldots, k\}, \forall v$.

In the description of the algorithm we will frequently need to express the fact that for every coloring c of G, for a pair of vertices x, y, either $c[x] = a$ or $c[y] = b$. We use the notation $\langle x = a \vee y = b \rangle$. A set of such clauses is maintained for every pair of vertices in the graph.

As a notational convenience, for $\mathbf{U} \subseteq \mathbf{V}$, we define $\mathbf{A}(\mathbf{U}) = \cup_{x \in \mathbf{U}} \mathbf{A}(x)$, the union of the available colors over $\mathbf{U}$. Thus, $\mathbf{A}(\mathbf{N}(v))$ is the set of available colors of the neighbors of v.

The algorithm proceeds in two stages.

Stage 1: In the first stage a recursive backtrack search reduces the graph by the deletion of edges and vertices and available colors, where each reduction step is recorded in a stack. When backtrack is forced, the stack is popped and the graph restored to its state at the previous search depth. If the graph becomes empty, the algorithm proceeds to the second stage, otherwise when no further options are available it reports failure.

Stage 2: In the second stage the stack information is used to reconstruct and color the graph, with each vertex assigned its final color as it is popped from the stack. In a final step, the coloring is verified for correctness on a copy of the original graph.

The graph reductions are generalizations and extensions of those in [5,10]. Many of the following reductions are applied recursively at each step of the search. Some may only be applied to the initial graph due to efficiency reasons, as early experiments inicated the overhead exceeded the savings in practice.

(1) If $\deg(v) < |\mathbf{A}(v)|$ then delete v.
(2) If $\mathbf{A}(v) \setminus \mathbf{A}(\mathbf{N}(v)) \neq \emptyset$ then delete v.

(3) If $\{u, v\} \in E$ and $\mathbf{A}(u) \cap \mathbf{A}(v) = \emptyset$ then delete edge $\{u, v\}$.
(4) If x, y, z is a 3-clique and $|\mathbf{A}(\{x, y, z\})| \leq 3$ then for each w which is a neighbor of 2 out of the three vertices, with $\mathbf{A}(w) = \mathbf{A}(\{x, y, z\})$ then merge w to the remaining vertex, say u. (deleting w). Also, remove from $\mathbf{A}(u)$ any color not in $\mathbf{A}(w)$.
(5) If $|\mathbf{A}(v)| = 1$ then delete v. For every $w \in \mathbf{N}(v)$, delete the 1 color in $\mathbf{A}(v)$ from $\mathbf{A}(w)$.
(6) If $\langle x = a \vee y = b \rangle$ and $\langle x = b \vee y = a \rangle$ then since $c[x] = c[y]$ merge y to x (deleting y). Remove from $\mathbf{A}(x)$ any color not in $\mathbf{A}(y)$.

In addition to reduction, each time a color is assigned to a vertex there is extensive forward propagation of the consequences. The propagation hinges on the clauses of the type $\langle x = a \vee y = b \rangle$. When a vertex is colored then that color is removed from the availsets of all adjacent vertices. The following may then be used to initiate such clauses.

- If $\{x, y\} \in \mathbf{E}$ and $\mathbf{A}(x) = \{a, b\}$, $\mathbf{A}(y) = \{a, c\}$ then we have $\langle x = b \vee y = c \rangle$.
- If $\{x, y\} \in \mathbf{E}$, and $\mathbf{A}(x) = \mathbf{A}(y) = \{a, b\}$ then we have both $\langle x = a \vee y = a \rangle$ and $\langle x = b \vee y = b \rangle$.
- If x, y, z is a triangle with $\mathbf{A}(\{x, y, z\}) = \{a, b, c\}$ and $\mathbf{A}(x) = \{a, b\}$ then $\langle y = c \vee z = c \rangle$.

Whenever a new clause is generated it may then interact with other clauses to produce more clauses. Note that the following does not depend on the edges of the graph; thus we refer to it as a *universal propagator*.

- For any three distinct vertices x, y, z and any colors a, b, c, d where $b \neq c$, if we have $\langle x = a \vee y = b \rangle$ and $\langle y = c \vee z = d \rangle$ then we obtain $\langle x = a \vee z = d \rangle$.

The primary use of propagators is to remove colors from availsets which prevents incorrect choices at later stages of the search.

- If $\langle x = a \vee y = a \rangle$ then $\forall w \in \mathbf{N}(x) \cap \mathbf{N}(y), a \notin \mathbf{A}(w)$.

Occasionally we can detect forced colorings directly from clauses. Note that once a vertex is colored then that color is removed from all its neighbors and we no longer need to consider the vertex as part of any clause.

- Forced Coloring: If $\langle x = a \vee y = b \rangle$ and $\langle x = a \vee y = c \rangle$ with $c \neq b$ then $\mathbf{A}(x) = \{a\}$.

The following indicates that we can limit the number of clauses to at most two per vertex pair. This is important for implementation, and it also gives us early detection of partial assignments that cannot be extended to a full coloring.

- If there are three pairwise non-equivalent relations of this form on a pair of vertices x and y, then either the coloring is inconsistent, or at least one of the vertices x or y has a forced coloring.

For each color choice, these rules are iterated until none apply or the current coloring is detected as inconsistent. The algorithm also uses dynamic variable ordering with heuristics based on degree and availsets.

The second stage, which constructs and verifies the coloring, is fairly straightforward and runs in $O(n^2)$ time.