ON THE THRESHOLD

Brian Hayes

A reprint from

American Scientist

the magazine of Sigma Xi, the Scientific Research Society

Volume 91, Number 1
January–February, 2003
pages 12–17

This reprint is provided for personal and noncommercial use. For any other use, please send a request to Permissions, *American Scientist*, P.O. Box 13975, Research Triangle Park, NC, 27709, U.S.A., or by electronic mail to perms@amsci.org. © 2003 Brian Hayes.

ON THE THRESHOLD

Brian Hayes

Last night I called technical support for the universe to report a bug. They kept me on hold for eternity, but finally I lodged my complaint: Some things in this world take entirely too long to compute—exponentially so, in the worst cases. "That's not a bug, that's a feature," was the inevitable reply. "It keeps the universe from running down too fast. Besides, NP-complete calculations are an unsupported option, which void your warranty. And where is it written that anything at all is guaranteed to be efficiently computable? Count yourself lucky that 1+1 is a polynomial-time calculation."

Perhaps cosmic tech support is right: Quick and easy answers to computational questions are not something we are entitled to expect in this world. Still, it's puzzling that some calculations are so much harder than others. The classic example is multiplication versus factoring. If you are given two prime numbers, it's easy to multiply them, yielding a bigger number as the product. But trying to undo this process—to take the product and recover the two unknown factors—seems to be much more difficult. We have fast algorithms for multiplying but not for factoring. Why is that?

Although such questions stump the help desk, there has been some progress lately in understanding the sources of difficulty in at least one family of computational tasks, those known as constraint-satisfaction problems. The new line of inquiry doesn't quite explain *why* some of these problems are hard and others are easy, but it traces the boundary between the two classes in considerable detail. Furthermore, a better map of the problem-solving landscape has led to a novel algorithm that pushes back a little further the frontier of intractability. The algorithm, called survey propagation, could well have important practical applications.

Where the Hard Problems Are

The new algorithm weaves together threads from at least three disciplines: mathematics, computer science and physics. The theme that binds them all together is the presence of sudden transitions from one kind of behavior to another.

The mathematical thread begins in the 1960s with the study of random graphs, initiated by Paul Erdős and Alfred Rényi. In this context a graph is not a chart or plot but a more abstract mathematical structure—a collection of vertices and edges, generally drawn as a network of dots (the vertices) and connecting lines (the edges). To draw a random graph, start by sprinkling n vertices on the page, then consider all possible pairings of the vertices, choosing randomly with probability p whether or not to draw an edge connecting each pair. When p is near 0, edges are rare, and the graph consists of many small, disconnected pieces, or components. As p increases, the graph comes to be dominated by a single "giant" component, which includes most of the vertices. The existence of this giant component is hardly a surprise, but the manner in which it develops is not obvious. The component does not evolve gradually as p increases but emerges suddenly when a certain threshold is crossed. The threshold is defined by a parameter I'll call α, which is the number of edges divided by the number of vertices. The giant component is born when α is about 1/2.

In computer science, a similar threshold phenomenon came to widespread attention in the early 1990s. In this case the threshold governs the likelihood that certain computational problems have a solution. One of these problems comes straight out of graph theory: It is the k-coloring problem, which asks you to paint each vertex of a graph with one of k colors, under the rule that two vertices joined by an edge may not have the same color. Finding an acceptable coloring gets harder as α increases, because there are more edges imposing constraints on each vertex. Again, the threshold is sharp: Below a certain α ratio, almost all graphs are k-colorable, and above this threshold almost none are. Moreover, the threshold affects not only the existence of solutions but also the difficulty of finding them. The computational effort needed to decide whether a graph is k-colorable has a dramatic peak near the critical value of α. (An influential paper about this effect was aptly titled "Where the *really* hard problems are.")

Physicists also know something about threshold phenomena; they call them phase transitions. But are the changes of state observed in random graphs and in constraint-satisfaction problems truly analogous to physical events such as the freezing of water and the onset of magnetization

Brian Hayes is senior writer for American Scientist. *Address: 211 Dacian Avenue, Durham, NC 27701; bhayes@amsci.org*

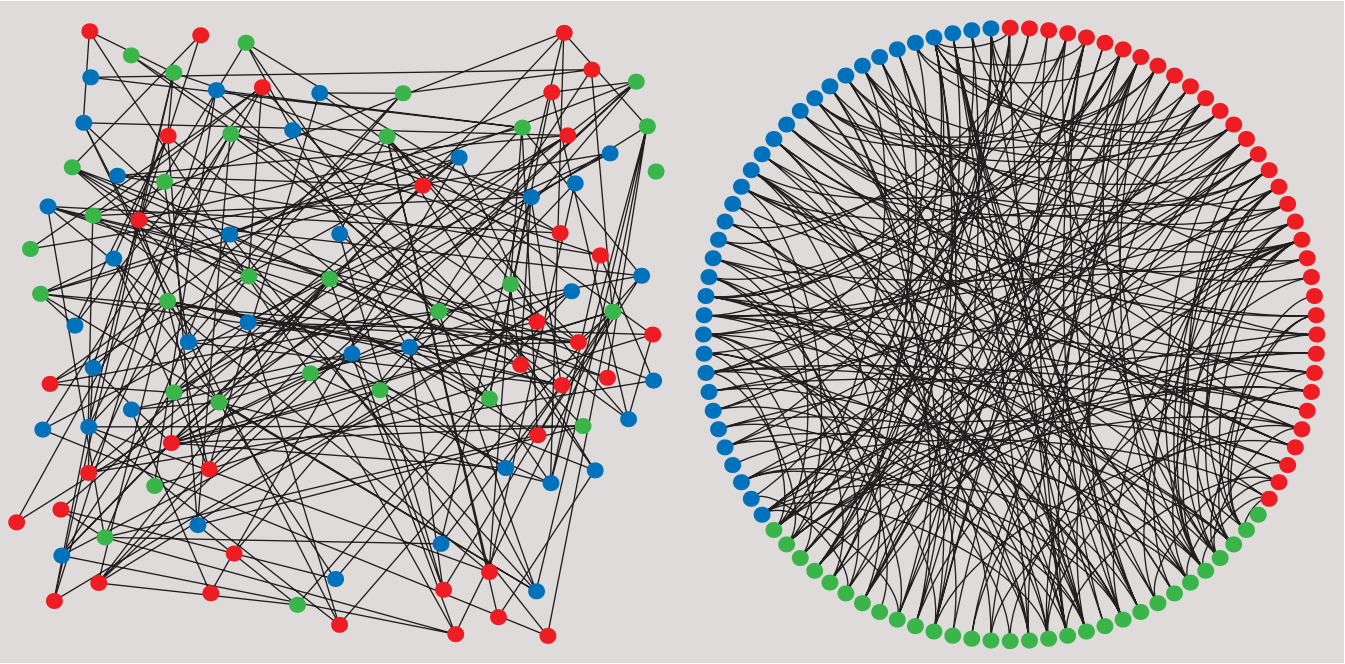

Figure 1. Graph coloring is one of several computational problems that have a sharp threshold where they get much harder to solve. A graph is a collection of vertices and edges, represented by dots and lines; the coloring problem is to assign each vertex a color—in this case chosen from a palette of just three—in such a way that no edge connects two vertices of the same color. When the ratio of edges to vertices is less than about 2.35, most random graphs can be colored in this way; above the threshold, very few can. The graph shown here has 100 vertices and 218 edges, and so it is a little below the threshold. A new algorithm called survey propagation has successfully colored graphs with up to a million vertices. The two diagrams above are views of the same graph; the circular arrangement at right makes it easier to verify that no edges link like-colored vertices.

in iron? Or is the resemblance a mere coincidence? For a time there was controversy over this issue, but it's now clear that the threshold phenomena in graphs and other mathematical structures are genuine phase transitions. The tools and techniques of statistical physics are ideally suited to them. In particular, the *k*-coloring problem can be mapped directly onto a model of a magnetic system in solid-state physics. The survey-propagation algorithm draws on ideas developed originally to describe such physical models.

Where the Hard Problems Aren't

Survey propagation is really a family of algorithms, which could be applied in many different realms. So far, the method has been tested on two specific problems. The first of these is Boolean satisfiability, or SAT, where the aim is to solve a large formula in symbolic logic, assigning values of *true* or *false* to all the variables in such a way that the entire formula evaluates to *true*. The second problem is *k*-coloring. Because I have written about satisfiability on an earlier occasion, I shall adopt *k*-coloring as the main example here. I focus on three-coloring, where the palette of available colors has just three entries.

Three-coloring is a hard problem, but not an impossible one. The question "Is this graph three-colorable?" can always be answered, at least in principle. Since each vertex can be assigned any of three colors, and there are n vertices, there must be exactly 3^n ways of coloring the graph. To decide whether a specific graph is three-colorable, just work through all the combinations one by one. If you find an assignment that satisfies the constraint—that is, where no edges yoke together like-colored vertices—then the answer to the question is yes. If you exhaust all the possibilities without finding a proper coloring, you can be certain that none exists.

This algorithm is simple and sure. Unfortunately, it's also useless, because enumerating 3^n colorings is beyond the realm of practicality for any n larger than 15 or 20. Some more-sophisticated procedures can retain the guarantee of an exact and exhaustive search while reducing the number of operations to fewer than 1.5^n. This is a dramatic improvement, but it is still an exponential function, and it merely raises the limit to $n=50$ or so. For large graphs, with thousands of vertices, all such brute-force methods are hopeless.

On the other hand, if you could somehow peek at the solution to a large three-coloring problem, you could check its correctness with much less labor. All you would have to do is go through the list of edges, verifying that the vertices at the ends of each edge carry different colors. The number of edges in a graph cannot be greater than n^2, which is a polynomial rather than an exponential function and which therefore grows much more slowly.

Problems with answers that are hard to find but easy to check are the characteristic signature of the class called NP (which stands for "nondeterministic polynomial"). Three-coloring is a charter member of NP and also belongs to the

more-elite group of problems described as NP-complete; the same is true of satisfiability. Barring a miracle, there will be no polynomial-time algorithms for NP-complete problems.

Having thus established the credentials of three-coloring as a certifiably hard problem, it is now time to reveal that most three-coloring problems on random graphs are actually quite easy. Given a typical graph, you have a good chance of quickly finding a three-coloring or proving that none exists. There is no real paradox in this curious situation. The classification of three-coloring as NP-complete is based on a worst-case analysis. It could be overturned only by an algorithm that is guaranteed to produce the correct answer and to run in polynomial time on every possible graph. No one has discovered such an algorithm. But there are many algorithms that run quickly most of the time, if you are willing to tolerate an occasional failure.

One popular strategy for graph-coloring algorithms is backtracking. It is similar to the way most people would attack the problem if they were to try coloring a graph by hand. You start by assigning an arbitrary color to an arbitrary vertex, then go on to the neighboring vertices, giving them any colors that do not cause a conflict. Continuing in this way, you may eventually reach a vertex where no color is legal; at that point you must back up, undoing some of your previous choices, and try again.

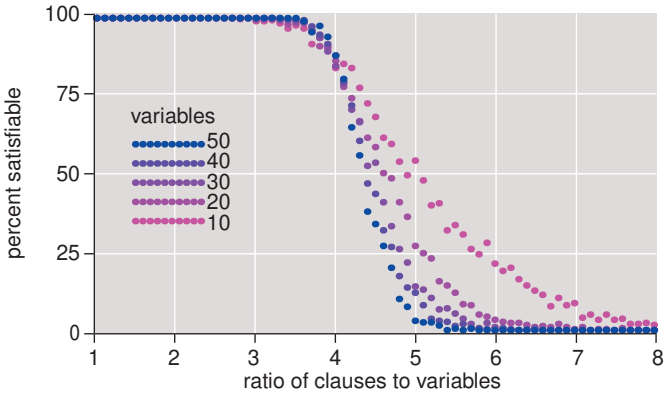

Figure 2. Transition between solvable and unsolvable phases is shown for the problem known as satisfiability, in which the task is to assign values of *true* or *false* to the variables in a logical formula made up of many clauses. As the ratio of clauses to variables increases, the problem changes from almost always solvable to almost never solvable. The transition gets sharper for larger problems.

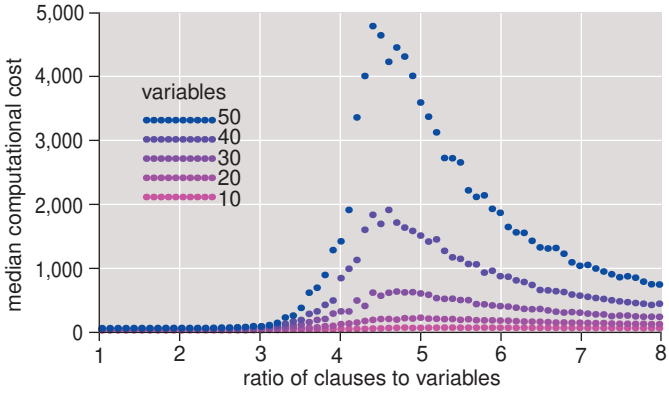

Figure 3. Computational effort needed to decide whether or not a formula is satisfiable has a peak near the phase transition. Problems well below the threshold are easy to solve, and those well above it are easily proved unsolvable; it's the ones in the middle that require patience. (The data in Figures 2 and 3 were previously published in Hayes 1997.)

Showing that a graph *cannot* be three-colored calls for another kind of algorithm. The basic approach is to search for a small cluster of vertices that—even in isolation from the rest of the graph—cannot be three-colored. For example, a "clique" made up of four vertices that are all linked to one another has this property. If you can find just one such cluster, it settles the question for the entire graph.

Algorithms like these are very different from the brute-force, exhaustive-search methods. The simple enumeration of all 3^n colorings may be impossibly slow, but at least it's consistent; the running time is the same on all graphs of the same size. This is not true for backtracking and other inexact or incomplete algorithms; their performance varies widely depending on the nature of the graph. In particular, the algorithms are sensitive to the value of α, the ratio of edges to vertices, which again is the parameter that controls the transition between colorable and uncolorable phases. Well below the critical value of α, where edges are sparse, there are so many ways to color the graph successfully that any reasonable strategy is likely to stumble onto one of them. At the opposite extreme, far above the threshold, graphs are densely interconnected, and it's easy to find a subgraph that spoils the chance of a three-coloring. The troublesome region is between these poles, near the threshold. In that middle ground there may be just a few proper colorings, or there may be none at all. Distinguishing between these two situations can require checking almost every possible assignment.

Where the Solutions Are

The critical value of α is about 2.35. In other words, if a random graph with n vertices has fewer than $2.35n$ edges, it can almost surely be three-colored; if it has more than $2.35n$ edges, a three-coloring is unlikely. Moreover, the transition between these two regimes is known to be sharp; it is a true discontinuity, a sudden jump rather than a smooth gradation. To put this idea more formally, the width of the transitional region tends to zero as n tends to infinity.

The sharpness of the phase transition could be taken as encouraging news. If algorithms for deciding colorability bog down only in the transitional region, and if that region is vanishingly narrow, then the probability of encountering a hard-to-classify graph is correspondingly small. But it seems the universe has another bug (or feature). In the first place, the sharpness of the colorability transition is assured only for infinitely

large graphs; at finite n, the corners of the transition curve are rounded. And there is another disrupting factor, which has been recognized only recently. It has to do not with the structure of the graph itself but with the structure of the set of all solutions to the coloring problem.

Although the uncolorable phase does not begin until $\alpha \approx 2.35$, experiments have shown that algorithms begin slowing down somewhat earlier, at values of α around 2.2. The discrepancy may seem inconsequential, but it is too large to be explained merely by the blurring of the phase transition at finite n. Something else is going on.

To understand the cause, it helps to think of all the possible three-colorings of a graph spread out over a surface. The height of the surface at any point represents the number of conflicts in the corresponding coloring. Thus the perfect colorings (those with zero conflicts) all lie at sea level, while the worst colorings create high-altitude peaks or plateaus. Of course the topography of this landscape depends on the particular graph. Consider how the surface evolves as α gradually increases. At low values of α there are broad basins and valleys, representing the many ways to color the graph perfectly. At high α the landscape is alpine, and even the lowest point is well above sea level, implying a complete absence of perfect colorings. The transitional value $\alpha \approx 2.35$ marks the moment when the last extensive areas of land at sea level disappear.

What happens in this "solution space" at $\alpha \approx 2.2$? It turns out this is the moment when a broad expanse of bottomland begins breaking up into smaller isolated basins. Below 2.2, nearly all the perfect colorings form a single giant connected cluster. They are connected in the sense that you can convert one solution into another by making relatively few changes, and without introducing too many conflicts in any of the intermediate stages. Above 2.2, each basin represents an isolated cluster of solutions. Colorings that lie in separate basins are substantially different, and converting one into another would require climbing over a ridge formed by colorings that have large numbers of conflicts. Algorithms that work by conducting a local search are unlikely to cross such ridge lines, and so they remain confined for long periods to whichever basin they first wander into. As α increases above 2.2, the number of perfect colorings within any one basin dwindles away to zero, and so the algorithms may fail to find a solution, even though many proper colorings still exist elsewhere on the solution surface.

This vision of solutions spread out over an undulating landscape is a familiar conceptual device in many areas of physics. Often the landscape is interpreted as an energy surface, and physical systems are assumed to run downhill toward states of minimum energy. This analogy can be pursued further, setting up a direct correspondence between the k-coloring of graphs and a model of magnetic materials.

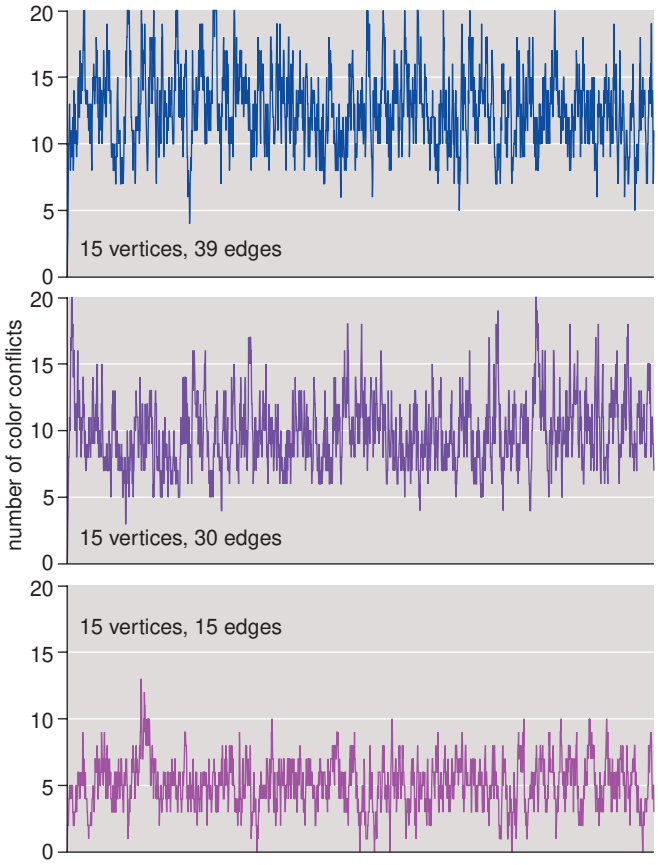

Figure 4. Random walks through the space of graph colorings offer some hints to why it's hard to find a perfect coloring. Each walk begins with a correct coloring and then repeatedly changes the color of a randomly chosen vertex. The traces record the number of color conflicts (edges connecting like-colored vertices) over 2,000 steps. Although all three graphs are known to be colorable, the random walk discovers perfect colorings only for the smallest graph *(bottommost trace)*.

Where the Spins Are

Models of magnetism come in baffling varieties. The basic components are vectors that represent atomic spins. Usually the spins are arranged in a regular lattice, as in a crystalline solid, and the vectors are constrained to point in only a few possible directions. In a model of a ferromagnet, nearby spins have positive couplings, meaning that the energy of the system is lower when the spins line up in parallel. An antiferromagnet has negative couplings, favoring spins that point in different directions. The problem of three-coloring a graph can be seen as a model of an antiferromagnet in which each spin has three possible directions, corresponding to the three colors. It is *anti*ferromagnetic because the favored state is one where the colors or the spins differ.

Most spin-system models focus on the effects of thermal fluctuations and the countervailing imperatives to minimize energy and to maximize entropy. In this respect the graph-coloring model is simpler than most, because the condition of interest is at zero temperature, where entropy can be neglected. On the other hand, the model is more complicated in another way: The spins are embedded in a graph with random interconnec-

tions, more like a glass than the geometrically regular lattice of a crystal.

Having translated the coloring problem into the language of spin physics, the aim is to identify the ground state—the spin configuration of minimum energy. If the ground-state energy is zero, then at least one perfect coloring exists. If the energy of the spins cannot be reduced to zero, then the corresponding graph is not three-colorable. The minimum energy indicates how many unavoidable conflicts exist in the colored graph.

Of course recasting the problem in a new vocabulary doesn't make the fundamental difficulty go away. In graph coloring, when you resolve a conflict by changing the color of one vertex, you may create a new conflict elsewhere in the graph. Likewise in the spin system, when you lower the energy of one pair of coupled spins, you may raise it for a different pair. Physicists refer to this effect as "frustration."

Interactions between adjacent spins can be viewed as a kind of message-passing, in which each spin tells its neighbors what they ought to do (or, since the coupling is antiferromagnetic, what they ought *not* to do). Translating back into the language of graph coloring, a green vertex broadcasts a signal to its neighbors saying "Don't be green." The neighbors send back messages of their own—"Don't be red," "Don't be blue." The trouble is, every edge is carrying messages in both directions, some of which may be contradictory. And feedback loops could prevent the network from ever settling down into a stable state.

A remedy for this kind of frustration is known in condensed-matter physics as the cavity method. It prescribes the following sequence of actions: First, choose a single spin and temporarily remove it from the system (thereby creating a "cavity"). Now, from among the neighbors surrounding the cavity, choose one node to regard as an output and consider the rest to be inputs. Sum up the signals arriving on all the input edges, and pass along the result to the output. The effect is to break open loops and enforce one-way communication. Finally, repeat the entire procedure with another spin, and continue until the system converges on some steady state.

The cavity method was first applied to constraint-satisfaction problems by Marc Mézard of the Université de Paris Sud, Giorgio Parisi of the Università di Roma "La Sapienza" and Riccardo Zecchina of the Abdus Salam International Centre for Theoretical Physics in Trieste. Initially it was a tool for calculating the average properties of statistical ensembles of many spin systems. About a year ago, Mézard and Zecchina realized that it could also be adapted to work with individual problem instances. But a significant change was needed. Instead of simple messages such as "Don't be green," the information transmitted from node to node consists of entire probability distributions, which give a numerical rating to each possible spin state or vertex color.

Mézard and Zecchina named the algorithm survey propagation. They got the "propagation" part from another algorithm that also inspired their work: a technique called belief propagation, which is used in certain error-correcting codes. "Survey" is meant in the sense of opinion poll: The sites surrounding a cavity are surveyed for the advice they would offer to their neighbors.

Where the Bugs Are

Over the past year the concept of survey propagation has been further refined and embodied in a series of computer programs by Mézard and Zecchina and a group of coworkers. Contributors include Alfredo Braunstein, Silvio Franz, Michele Leone, Andrea Montanari, Roberto Mulet, Andrea Pagnani, Federico Ricci-Tersenghi and Martin Weigt.

To solve a three-coloring problem on a graph of size n, the algorithm first finds the vertex that is most highly biased toward one color or another, and permanently sets the color of that vertex accordingly. Then the algorithm is invoked recursively on the remaining graph of $n-1$ vertices, so that another vertex color is fixed. Obviously this process has to terminate after no more than n repetitions. In practice it usually stops sooner, when all the signals propagating through the network have become messages of indifference, putting no constraints on neighboring nodes. At this point survey propagation has nothing more to offer, but the graph that remains has been reduced to a trivial case for other methods.

As with other algorithms for NP-complete problems, survey propagation comes with no guarantees, and it does sometimes fail. The process of deciding which vertex to fix next is not infallible, and when a wrong choice is made, there may be no later opportunity to recover from it. (Adding some form of backtracking or randomized restarting might alleviate this problem.) In its present form the algorithm is also strictly one-sided: It can usually color a colorable graph, but it cannot prove a graph to be *un*colorable.

Nevertheless, the algorithm has already had some impressive successes, particularly in the hard-to-solve region near the phase transition. The version for satisfiability has solved problems with 8 million variables. The graph-coloring program handles graphs of a million vertices. Both of these numbers are two orders of magnitude beyond what is routine practice for other methods.

Graph coloring and satisfiability are not just toy problems for theorists. They are at the core of various practical tasks in scheduling, in the engineering of silicon circuits and in optimizing computer programs. Having an algorithm capable of solving much larger instances could open up still more applications.

Ironically, although survey propagation works well on enormous problems, it sometimes stalls on much smaller instances, such as random graphs with only a few hundred vertices. This is

not a pressing practical concern, since other methods work well in this size range, but it's annoying, and there's the worry that the same failures might show up in larger *non*random graphs. The cause of these small-graph failures is not yet clear. It may have to do with an abundance of densely nested loops and other structures in the graphs. Then again, it may be just another bug in the universe.

Acknowledgment

This article had its genesis during a 10-week residence at the Abdus Salam International Centre for Theoretical Physics, where I benefitted from discussions with Riccardo Zecchina, Muli Safra, Roberto Mulet, Marc Mézard, Stephan Mertens, Alfredo Braunstein, Johannes Berg and others.

Bibliography

Achlioptas, Dimitris, and Ehud Friedgut. 1999. A sharp threshold for *k*-colorability. *Random Structures and Algorithms* 14:63–70.

Beigel, Richard, and David Eppstein. 2000. 3-coloring in time $O(1.3289^n)$. http://arXiv.org/abs/cs/0006046

Braunstein, A., Marc Mézard and Riccardo Zecchina. 2002. Survey propagation: An algorithm for satisfiability. Manuscript in preparation.

Cheeseman, Peter, Bob Kanefsky and William M. Taylor. 1991. Where the *really* hard problems are. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 331–337.

Dubois, O., R. Monasson, B. Selman and R. Zecchina (eds). 2001. Special issue on phase transitions in combinatorial problems. *Theoretical Computer Science* 265(1).

Erdős, P., and A. Rényi. 1960. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences* 5:17–61.

Friedgut, Ehud, and Gil Kalai. 1996. Every monotone graph property has a sharp threshold. *Proceedings of the American Mathematical Society* 124:2993–3002.

Gent, Ian P., and Toby Walsh (eds). 2002. Satisfiability in the year 2000. *Journal of Automated Reasoning* 28(2).

Hayes, Brian. 1997. Computing science: Can't get no satisfaction. *American Scientist* 85:108–112.

Johnson, David S., and Michael A. Trick (eds). 1996. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. Providence, R.I.: American Mathematical Society.

Martin, Olivier C., Rémi Monasson and Riccardo Zecchina. 2001. Statistical mechanics methods and phase transitions in optimization problems. *Theoretical Computer Science* 265:3–67.

Mézard, Marc, Giorgio Parisi and Miguel Angel Virasoro (eds). 1987. *Spin Glass Theory and Beyond*. Philadelphia: World Scientific.

Mézard, Marc, and Giorgio Parisi. 2002. The cavity method at zero temperature. *Journal of Statistical Physics* (in press).

Mézard, M., G. Parisi and R. Zecchina. 2002. Analytic and algorithmic solution of random satisfiability problem. *Science* 297:812–815.

Mézard, Marc, and Riccardo Zecchina. 2002. Random 3-SAT: From an analytic solution to a new efficient algorithm. *Physical Review E* (in press).

Mulet, R., A. Pagnani, M. Weigt and R. Zecchina. 2002. Coloring random graphs. *Physical Review Letters* (in press).

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