

Phase Transitions and Algorithmic Thresholds

David W. Etherington
CIRL
1269 University of Oregon
Eugene OR 97404
USA

<http://www.cirl.uoregon.edu/ether/>

Andrew J. Parkes
CIRL
1269 University of Oregon
Eugene OR 97404
USA

<http://www.cirl.uoregon.edu/parkes/>

ABSTRACT

Summary of the results on thresholds in a particular anytime algorithm based on (distributed) iterative repair. Discussion of how such thresholds are algorithm dependent in contrast to algorithm-independent phase transitions.

1. INTRODUCTION

Everyone is familiar with the way that the properties of physical systems can change abruptly as we vary their temperature. The properties of H₂O at -1°C (ice) is quite different from that at +1°C (liquid water), whereas, a 2°C change makes little difference at other temperatures until we reach 99°C. Thus as we vary the control parameter, temperature, the properties exhibit phases: e.g., ice, water, steam. Inside a phase, the properties of the systems change slowly, but there are abrupt changes between phases.

The abruptness of such changes is due to the large number of degrees of freedom within the system—e.g., the large number of water molecules. Systems of small numbers of molecules do not show phase transitions. That is, the statistical properties of very large systems can show effects that are not apparent in small systems. We are familiar with this for physical systems, but it can also happen for non-physical systems, and in particular for systems of interacting constraints. Hence, since such systems are relevant to optimization problems, to decision and optimization problems.

An example that we often use in the studies is that of a phase transition in satisfiability problems, and specifically that of Random 3SAT, a problem that has been extensively studied. In this case the control parameter, t , is the ratio of clauses to variables, and the property, P , that we care about is whether or not the instance has a satisfying assignment. Given an algorithm to determine satisfiability the typical behavior is shown in Figure 1, with a phase transition at $t_C \approx 4.2$

The important aspect here is that the phase transition associated with NP-hard properties, such as satisfiability, typically splits up the parameter space into 3 regions

- **easy.** $t < t_C$. Instances are almost always satisfiable, that is, the chance that they are unsatisfiable is “infinitesimal”¹. Furthermore, it is relatively easy to find a solution.
- **hard.** $t \approx t_C$ these critical instances have finite probability of being sat, and the same for unsat. This region is the hardest to solve, because instances, and the sub-problems that

¹Of course, this all has a precise statement which can be found in the literature

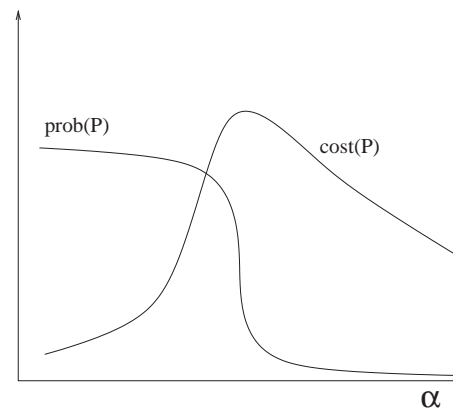


Figure 1: Typical behavior at a phase transition of some property “ P ” such as “satisfiable.” The x-axis is the value of some control parameter, t , for example, the clause/variable ratio. The line marked “ $\text{prob}(P)$ ” is the probability that P is true in some randomly selected instance at t . The line “ $\text{cost}(P)$ ” is the average search cost of determining whether or not P holds true in an instance.

arise during search, are always “on the edge between sat and unsat” and distinguishing on which side they lie is hard.

- **less-hard.** $t > t_C$ (often inaccurately called “easy” again). Here problems are almost always unsatisfiable. Solving them is much easier than for the critical region, but still (typically) significantly harder than for the “easy” region.

The important aspect here is that the peak in the search cost seems to apply to many (possibly all) algorithms. It is algorithm independent and is associated with a transition in the properties of the instances.

Some of our work has been directed at refining this picture from the point of view of picking out and clarifying aspects that are relevant to ANTS, that is to search that might be distributed, but that tends to focus on obtaining fast solutions in the “easy” region.

We discuss each of our contribution in turn. Firstly, [2] looks at whether the easy region is as easy as one might hope it should be for distributed algorithms. The paper [3] attempts to quantify the meaning of “easy.” Then, [4], follows up and shows a surprising change in scaling behavior that occurs within the easy region, and away from the phase transition. We will finish this high-level review of CIRL work with a discussion of the import of these results for ANTS systems.

2. COMPLEXITY OF DISTRIBUTED LOCAL SEARCH

This paper [2] asks the question:

As we move away from the phase transition region into the underconstrained region, do the problems become efficiently solvable, on average, by parallel (as well as sequential) algorithms?

For a sequential algorithm, “efficient” usually means polynomial in the number of variables ($poly(n)$). However, for a parallel algorithm, “efficient” we want to take good advantage of parallel computation, and so is generally taken to mean being polynomial in the log of the number of variables ($poly(\log(n))$) or “polylog time”.

The question is non-trivial because there are problems (called P-complete) that can be solved efficiently sequentially, but are generally believed to have no efficient solution. It could have happened that the easy problems below the phase transition had no efficient parallel solution, and this would have had bad implications for the underpinnings of ANTS. However, in fact we found (empirically) that they did have efficient parallel solutions.

In particular, we studied the behavior of a distributed local search algorithm, WSAT, since such algorithms are a fairly common method to solve distributed optimization problems. On Random 3SAT, WSAT exhibited polylog scaling, specifically we found that distributed WSAT is $O(\log(n^2))$.

3. SCALING OF WALKSAT

In [3] we study the scaling over the entire easy region. In comparison most previous studies have been restricted to just the phase transition region. Over most of this region the scaling of the (sequential) local search algorithm WalkSAT is found to be a simple power-law.

The final law (equation 12 of [3]) is that the number of flips needed scales as:

$$\frac{g_0 n}{(\alpha_0 - \alpha)^{g_1}} \quad (1)$$

where α is the clause/variable ratio, and g_0, g_1 and α_0 are constants.

The utility of this simple scaling law is to suggest that the scaling behavior of even difficult-to-predict algorithms can empirically have good behaviors.

Initial intentions are:

- scaling laws might prompt some theoretical work—it seems reasonable that it ought to be predictable in the easy region
- provide an example of an empirical law, that might be useful in other domains, and so might be useful for empirical predictions of scaling behavior

Note that this formula is valid only for $\alpha < \alpha_0$, but that for any such α that the scaling is linear.

However, the surprise was that experiments yielded that $\alpha_0 \approx 4.19$ was significantly below the phase transition point. This suggested that there could be a significant change in the scaling behavior at a point distinctly below the phase transition.

4. SCALING OF PURE WALKSAT

Unfortunately, although [3] suggested the existence of a threshold below the phase transition, the algorithm used was so effective that the potential threshold was so close to the phase transition that separating it from the phase transition was experimentally difficult.

Hence, in [4] we studied an algorithm that was heuristically less effective, and so likely to have a threshold further from the phase transition. We also selected for study an algorithm that is very simple and so might (we hope) ultimately be amenable to theoretical study.

The algorithm used is that of “Pure Pure Random Walk” (PRWSAT)

```

P := a randomly generated truth assignment
for j := 1 to MAX-FLIPS {
  if ( P is a solution ) then return P
  else
    c := a randomly selected unsatisfied clause
    flip a random literal from c }
return failure

```

This algorithm [1] predates (and inspired) the usual WalkSAT. Somewhat remarkably, despite its naivete, it is average polytime on 2SAT (which has a linear time algorithm, but it is still somewhat surprising that such a naive local search algorithm does not end up making exponentially long mistakes).

We find experimental indications of a threshold at $\alpha \approx 2.7$

- $\alpha < 2.7$ The scaling of PRWSAT is simple linear. Furthermore, the variance of runtimes between different instances is small. That is, in this region, the runtime of the local search algorithm, is not only fast, but is also highly predictable between instances.
- $\alpha > 2.7$. In this region the scaling appears to be bad (not linear, and possibly not even polynomial). Furthermore, the variance of runtime between instances is no longer small. That is, the scaling is bad, and there is also little ability to predict the runtime on an instance.

Note that the threshold is different than for the heuristically-enable algorithm studied earlier. That is, the position of the threshold is algorithm dependent, in contrast to the algorithm independence of the phase transition itself.

That is, we have a threshold, depending on the algorithm, that separates regions

- **good** linear or polytime scaling, low variance between instances. Runtimes are low and predictable
- **bad** bad (exponential) scaling, high variance between instances. Runtimes are high and unpredictable.

Clearly, for ANTS systems, in which we want to be able to predict runtimes, we want to be below the threshold for the algorithm.

5. CONCLUSIONS

Our belief before this work—a reflection of the general belief of the community—was that, while the algorithms slow down as we approach the phase transition, that it would do so gradually. Hence, the main lesson learned was that sharp transitions, from good scaling to bad scaling properties can also occur when using a local search method. Furthermore the location of the threshold can vary with the details of the algorithm—in contrast to the phase transition itself which is algorithm independent.

Better algorithms presumably have a threshold that is closer to the phase transition, but will require more computational effort.

Hence there is a tradeoff:

- high quality algorithm - slower, but might be a net win because end up in the poly scaling region

- low quality algorithm—faster, if the solution quality expected to be achievable in the time limit is low enough, then we might effectively be in the easy region, and so the resulting poly scaling means the algorithm will produce good enough results soon enough—might be a net win, because of the reduced communication costs in implementing the simpler algorithm

A reasonable goal would be to estimate the solution quality likely to be achievable in the time limit, and then use this to select the cheapest algorithm that would be in its easy region at that quality.

6. ACKNOWLEDGMENTS

This was sponsored in part by grants from the Defense Advanced Research Projects Agency (DARPA), under contract number F30602-00-2-0534. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, Rome Laboratory, or the U.S. Government.

7. REFERENCES

- [1] C. Papadimitriou. On selecting a satisfying truth assignment. In *Proc. IEEE symposium on Foundations of Computer Science*, pages 163–169, 1991.
- [2] A. J. Parkes. Distributed local search, phase transitions, and polylog time. In *Proceedings of the workshop on “Stochastic Search Algorithms”, held at “Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)”*, August 2001. <http://www.cirl.uoregon.edu/parkes/>.
- [3] A. J. Parkes. Easy predictions for the easy-hard-easy transition. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, July 2002. <http://www.cirl.uoregon.edu/parkes/>.
- [4] A. J. Parkes. Scaling properties of pure random walk on random 3-SAT. In *Proceedings of Eighth International Conference on Principles and Practice of Constraint Programming (CP 2002)*, volume LNCS 2470 of *Lecture Notes in Computer Science*, pages 708–713, 2002. <http://www.cirl.uoregon.edu/parkes/>.