SAT Distributions with Phase Transitions between Decision and Optimization Problems

Tassos Dimitriou

Athens Information Technology,
Markopoulo Ave., 190 02, Peania, Athens, Greece.

Abstract

We present a generator for *weighted* instances of MAX k-SAT in which every clause has a weight associated with it and the goal is to maximize the total weight of satisfied clauses. Our generator produces formulas whose hardness can be finely tuned by two parameters p and δ that control the weights of the clauses. Under the right choice of these parameters an easy-hard-easy pattern in the search complexity emerges which is similar to the patterns observed for traditional SAT distributions.

What is remarkable, however, is that the generated distributions seem to lie in the middle ground between decision and optimization problems. Increasing the value of p from 0 to 1 has the effect of changing the shape of the computational cost from an easy-hard-easy pattern which is typical of *decision* problems to an easy-hard pattern which is typical of *optimization* problems. Thus our distributions seem to bridge the gap between decision and optimization versions of SAT.

Furthermore, we demonstrate that these phase transitions are related to sudden changes to a quantity similar to the backbone of a SAT formula. In our model not only we know how the optimal solution looks like but we also prove it is *unique*. Thus our generator comes with a proof of the optimality of the best assignment which is basically the structural property that is related to the phase transition phenomena observed.

1 Introduction

Phase transition phenomena in combinatorial search problems have proved a fertile source of research activity for over a decade. An informal description of a "phase transition" is the behavior whereby "small" changes in certain parameters of a system cause dramatic shifts in some globally observed quantity.

Email address: `tdim@ait.gr` (Tassos Dimitriou).

A typical example of such a behavior is the satisfiability (SAT) of Boolean formulas. The computational cost of solving random 3-SAT instances (formulas in Conjunctive Normal Form with 3 literals per clause) exhibits transitions from easy to hard and back to easy[10,6] as the ratio of number of clauses to variables increases. In combinatorial graph theory, similar phenomena have been observed with respect to random n-vertex graphs in which edges are added with some probability $p(n)$; when one considers a certain property Π (connectivity, 3-colorability, etc.) then there is a value for the edge probability $p(n)$ where the property Π appears abruptly[5,1].

The interest in phase transition phenomena stems from experimental studies of search heuristics for NP-complete problems[3,8,10,6,7], where the probability of a random instance having a solution is mirrored in the run-time behavior of the methods used to find the solution. Phase transitions usually depend on some control or order parameter that can be adjusted to control the hardness of the problem. For example, the probability that a random graph is connected or has a hamilton cycle, etc. depends on the edge density[5,1]; the satisfiability and the hardness of 3-SAT formulas depends on the ratio of clauses to variables [3,10,4,6], and so on.

In particular, instances "outside" the threshold region are typically solved easily as opposed to instances close to the threshold point which are much harder to solve. Furthermore, it has been observed that phase transitions for NP-complete decision problems have easy-hard-easy patterns while phase transitions for the corresponding optimization problems follow easy-hard patterns[7,15,14].

Our motivation for this research is threefold; First we want to introduce a new distribution of SAT instances that will bridge the gap and possibly help us understand the relationship between the phase transitions of decision problems and those of their optimization counterparts. Second, we want to identify and locate difficult instances that can be used in the development of new solving methods. Finally, we want to understand the characteristics of optimal solutions and the behavior of algorithms for finding them with respect to certain structural properties of the instances at hand.

The instances we generate are k-SAT formulas where every clause has a *weight* associated with it. The goal is to find an assignment that maximizes the *sum of weights* of the satisfied clauses. The generated instances are parameterized by two quantities p and δ which control the weights associated with the clauses. By carefully setting the values of these two parameters one can generate difficult to solve formulas, thus making it possible to test algorithms on *hard* generated instances only.

Furthermore, our generator has two more important characteristics that are related to the values of p and δ. The first one is a theoretical result proving that the optimal assignment is *unique*. Since any satisfiability heuristic when fed with an instance from our generator will try to maximize the weight of satisfied clauses, this characterization provides algorithm designers with an *a priori* knowledge of the optimal assignment. We call this solution the *hidden* or *planted* assignment. Thus by knowing what to expect, algorithm designers will be able to evaluate better the effectiveness of their algorithms.

The second characteristic is the appearance of an easy-hard-easy pattern in the search complexity for the optimal assignment. Although the problem we consider here is a *maximization* one and phase transitions should exhibit "easy-hard" patterns[7,15,14], by increasing the value of p from 0 to 1 one starts with "easy-hard-easy" patterns which are typical of *decision* problems to end up with "easy-hard" patterns which are typical of *optimization* problems. Thus our distributions seem to bridge the gap between decision versions and optimization versions of SAT. Furthermore, we were able to link this behavior with a new threshold phenomenon which is related to the uniqueness of the hidden assignment. Below the threshold, there are other solutions that achieve equal total weight and differ from the hidden one in a few variables. Above the threshold however, the hidden assignment becomes the unique optimal solution. Thus there exists a transition from a phase where there are more than one good assignments to a phase where the optimal assignment is unique. The point to be made is that this transition coincides with the hardest to solve problem instances.

2 Generator for MAX k-WSAT

Our generator produces *weighted* instances of the MAX k-SAT problem, which we call MAX k-WSAT. In general, MAX k-WSAT consists of Boolean expressions in conjunctive normal form, i.e. collection of clauses in which every clause consists of exactly k literals and has a positive integer weight associated with it. Given an instance of this problem, one is looking for an assignment to the variables that satisfies a set of clauses with maximum total weight.

It is clear that MAX k-WSAT is NP-hard as MAX k-SAT reduces to it by setting all weights equal to one. In this work we will present a generator for instances for a degenerate version of MAX k-WSAT, in which *all* weights to the clauses are either n^2 or $n^2 + 1$, where n is defined below. While this simplification may seem very restrictive at first look, it is all we need to create a generator of k-SAT instances with useful computational properties. Furthermore, even when $k = 2$, the problem remains NP-hard.

To generate a formula with the above properties we first start with $2n$ vari-

The model $F_{n,p,\delta}$ (with super-clauses)

(1) Start with $2n$ variables, n green and n blue.
(2) **(Create the formula)** For every 2-tuple of variables x_1, x_2, irrespective of their color and without repetitions, add to the formula the "super-clause"

$$c(x_1, x_2) = \neg(x_1 x_2 + \bar{x}_1 \bar{x}_2).$$

(3) **(Assign the weights)** For all clauses $c(x_1, x_2)$, set the clause weight $w(x_1, x_2)$ according to the following rule: If x_1, x_2 have the same color then

$$w(x_1, x_2) = \begin{cases} n^2 + 1, & \text{with probability } p \\ n^2, & \text{otherwise} \end{cases}$$

If x_1, x_2 have different colors then

$$w(x_1, x_2) = \begin{cases} n^2 + 1, & \text{with probability } p + \delta \\ n^2, & \text{otherwise} \end{cases}$$

Fig. 1. Generator for 2-WSAT formulas.

ables, n green and n blue, create the clauses and finally assign weights to them. We call our model $F_{n,p,\delta}$, where n indicates the number of variables of each color and p, δ are the parameters used to control the maximum total weight achieved by the hidden assignment (Figure 1). The user can choose any values for δ and p provided $p+\delta \leq 1$. (While we only show the generator for 2-WSAT formulas, the extension to k-WSAT formulas should be straightforward.)

By looking at Figure 1 one should observe that the "clauses" $c(x_1, x_2)$ are not really clauses in the ordinary 2-SAT sense. In fact, $c(x_1, x_2) = (x_1+x_2)(\bar{x}_1+\bar{x}_2)$. We chose, however, to work with super-clauses as the results are much easier to describe and the passing to ordinary 2-SAT expressions is again easy.

It is also clear from the model that the generated formulas are "dense" in that they consist of all possible combinations of the $2n$ variables. Thus it makes no sense to try to satisfy all super-clauses but it makes sense to try to satisfy a suitable subset of those that incurs the maximum possible total weight. We will be able to show later on (Theorem 3) that the best assignment (the *hidden assignment* as we call it) is the one that has the green variables set to true and the blue set to false (or vice versa).

Definition 1 *An assignment is said to split the variables if exactly n variables* *are set to true and n are set to false (irrespective of their color).*

The next lemma is used to reduce the space of good assignments. Since our goal is to generate formulas where assignments are planted, this lemma allows algorithm designers to test the quality of solutions found by their algorithms by knowing what to expect for.

Lemma 2 (Look for split assignments) *Any assignment that is not evenly split is outweighted by some assignment with split variables.*

Proof (Sketch) The particular choice of weights assigned to clauses makes the overall weight achieved by the *best* unevenly split assignment *less* than the weight achieved by *any* assignment with split variables. Thus it is always best to look for split assignments. □

While so far we have considered only WSAT formulas with super-clauses, the same property applies when we break each super-clause to its constituent clauses. We simply have to modify the model by assigning the weight $w(x_1, x_2)$ to each of the two sub-clauses. Thus from now on we will work only with formulas that consist of super-clauses. To simplify things further we will work only with split assignments since by Lemma 2 we are allowed to do so.

Our goal now is to show that for a suitable choice of the parameter δ, the optimal assignment is one that has the green variables set to True and the blue variables set to False (or vice versa).

Theorem 3 (Optimality of hidden assignment) *There is a constant such that for values of $\delta \geq \Omega(\sqrt{(1-p)\ln n/n})$, the assignment which has only the green variables set to true is optimal with high probability.*

Proof (Sketch) The idea is to use Chernoff bounds to show that any split assignment that has not separated the green from the blue variables has a neighboring assignment that achieves even better weight, except of course the hidden assignment. This suggests that these assignments cannot be optimal. Eventually, we get that the hidden assignment is optimal with high probability for the range of δ described in the theorem. □

The optimality theorem characterizes implicitly the values of p for which it is safe to assume that the hidden assignment is optimal with high probability.

Since by the definition of the model we know that δ must be less than $1-p$, it is clear that the theorem will be true for values of p satisfying

$$1-p \geq \Omega(\sqrt{(1-p)\ln n/n})$$

or equivalently

$$p \leq 1 - c\frac{\ln n}{n} \tag{1}$$

for some constant c. Thus our approach cannot be used for all formulas, but only for formulas where the monochromatic clauses are not too heavy, as indicated by Equation 1 and the definition of the model.

3 Hardness Results for WSAT

Our motivation in this section is to show that easy and hard k-WSAT instances can be predictable in advance. This will enable designers of local search SAT heuristics to test their algorithms on hard k-SAT instances only in which the optimal solution is known beforehand. In the experiments that follow we chose to work with MAX 2-WSAT formulas to illustrate the fact that these formulas become extremely difficult to optimize in direct contrast to ordinary 2-SAT formulas, which are solvable in linear time[2]. Although we leave a more detailed analysis of k-WSAT formulas, $k \geq 3$, for the final version of this paper, preliminary work shows that they exhibit similar properties to the 2-WSAT case. In all the figures that follow each sample point was computed after generating 1000 random instances of MAX 2-WSAT.

The local search procedure we used for our tests is a modified version of WalkSat[12] in which the strategy we used is to flip the variable that belongs to a random unsatisfied clause and results in the smallest decrease in the overall *weight* of satisfied clauses. The main reason for choosing WalkSat is because it is one of the best performing SAT procedures and because we believe that these results on hard instances will be applicable to other SAT heuristics as well. In the final version of this paper we plan to conduct experiments with other heuristics and complete methods as well.

Figure 2 shows the *median* of the *total number of variable flips* required by WalkSat to locate an assignment that achieves the maximum total weight (as is implied by the hidden assignment) for 2-WSAT formulas with p equal to $\frac{1}{2}$ and $n = 32, 34, 36, 38$ and 40. As can be seen, an easy-hard-easy pattern emerges which results in an exponential increase in computational cost in the hardest region similar to the behavior of ordinary 3-SAT formulas [10,6].

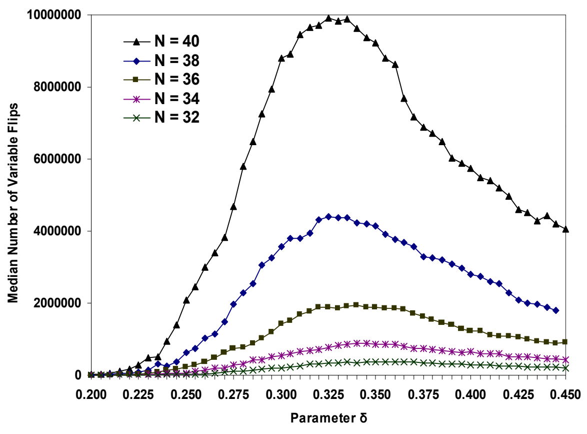

Fig. 2. Median number of total variable flips for random 2-WSAT formulas as a function of the parameter δ, when $p = 1/2$.

It is perhaps instructive at this point to comment a little on the shape of the curves in Figure 2. Although the computational cost follows an easy-hard-easy pattern, the second "easy" region where δ is large is no longer very easy compared to the first region where δ is small. This is reminiscent of the behavior of 3-SAT(B), the *bounded decision* versions of 3-SAT defined by Zhang[14], where one is looking for an assignment that violates no more than B constraints. When $B = 0$, one has 3-SAT; when B is the optimal solution cost, one has MAX 3-SAT. Thus, such distributions lie in some sense between the decision problem and its optimization counterpart and like the WSAT instances exhibit easy-hard-"less easy" patterns.

In general, as was shown in [7,15,13,14] and other works, the phase transitions of some NP-complete *decision* problems follow easy-hard-easy patterns and the phase transitions of some NP-hard *optimization* problems follow easy-hard patterns. Thus one may ask, where is the easy-hard behavior of the WSAT formulas? As we will see in Figure 3, WSAT formulas exhibit the behavior of optimization problems but only when p grows larger than 1/2. Thus indeed the value of $p = 1/2$ is middle ground and by increasing the value of p one gets a wealth of distributions with higher computational costs.

Figure 3 shows the computational cost required to find a good assignment for 2-WSAT formulas with $n = 34$ variables and p ranging from 0.1 to 0.8. (Similar findings for other values of n are omitted due to space restrictions.) Starting with $p = 0.1$ (curve in the front) we see that the point of maximum cost is moving slowly to the right with a parallel increase in its maximum value, as p becomes 0.4. However, when p becomes 0.5 or larger the points of maximum cost are moving slowly to the left to acquire the maximum value when $p = 0.8$. All the curves exhibit easy-hard-"less easy" patterns with the exception of the curve for $p = 0.8$ which has an easy-hard pattern as δ increases from 0 to its maximum value 0.200. Thus in this particular curve the computational cost

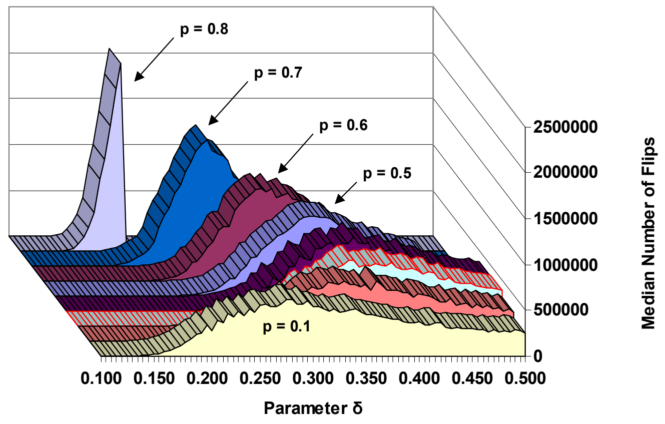

Fig. 3. Computational cost for random 2-WSAT formulas with $n = 34$ and various values of p and δ.

remains maximum for values of $\delta > 0.200$.

4 Phase Transitions

An important characteristic of Figure 2 is that the transition region becomes narrower (occurs for a smaller range of δ) for larger values of n when at the same time the peak shifts to the left as n is increased.

Our goal in this section is to demonstrate a relationship between the hard region and a phase transition in the structural properties of the WSAT formulas. It is clear that we cannot have a SAT/UNSAT transition as all instances are unsatisfiable. A more profound concept related to phase transitions is that of a *backbone* which in some sense is the set of all *frozen decisions*[11,13], i.e. those with fixed outcomes for all possible solutions. For example, in SAT the backbone of a formula is the set of all literals that are true in all satisfying assignments[11]. A phase transition in such a case has the backbone ratio raise from nearly 0 to nearly 1, with the hardest instances lying around the 50% point, not only in their decision version but in their optimization as well[11,13–15]. In the case of WSAT formulas, however, we chose not to work with backbones as there is essentially only one solution and most of the variables have a fixed value. We were able, however, to relate the WSAT behavior with the probability of uniqueness of the hidden assignment, which is the crucial structural property of WSAT formulas.

Figure 4 shows the uniqueness probability of the optimal solution for $p = 1/2$ and a large range of values for n. Observe how the threshold function sharpens up for larger values of n, like the satisfiability threshold function for random k-SAT formulas[10]. So, now, the question becomes: given an arbitrary value

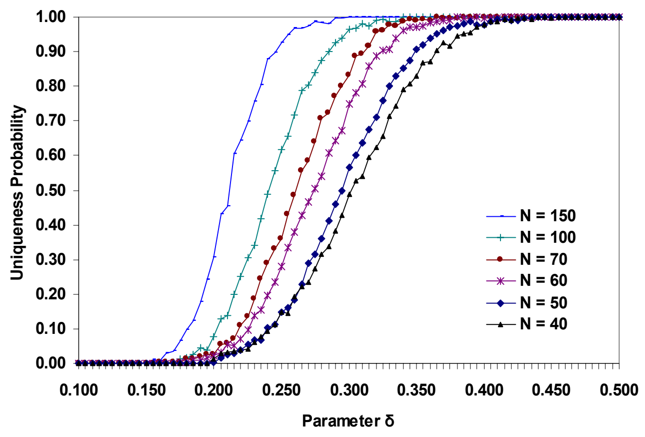

Fig. 4. Phase transition for $p = 1/2$ and various values of n.

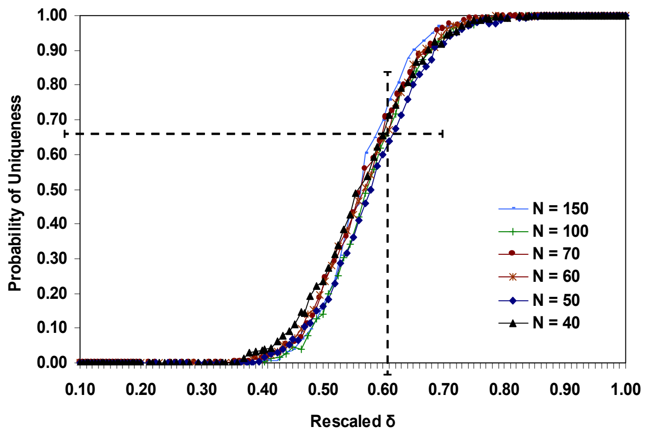

Fig. 5. Phase transition for $p = 1/2$ and various values of n, after rescaling.

of n how can we determine the value of δ that results in the most difficult to solve instances? The answer is given by *finite-size scaling*[9], in which the horizontal axis is rescaled by a quantity that is a function of n.

Figure 5 shows the result of rescaling the curves of Figure 4. The uniqueness probability is plotted against δ', a rescaled version of δ equal to $\delta' = \delta n^{\epsilon/2}\sqrt{1-\epsilon}$, where $\epsilon = 0.56$. Finally, Figure 6 demonstrates how the computational cost for various values of n collapses into a universal curve. To obtain these data we first normalized the curves shown in Figure 2 and then applied the rescaling described previously. We see clearly that the critical point is when the *rescaled* δ is equal to 0.60 which corresponds to the 65% uniqueness probability in Figure 5. Thus the main empirical observation we can draw from these pictures is that when $p = 1/2$, *the hardest 2-WSAT formulas lie at the point where about 65% of them have the hidden assignment as the optimal one.*

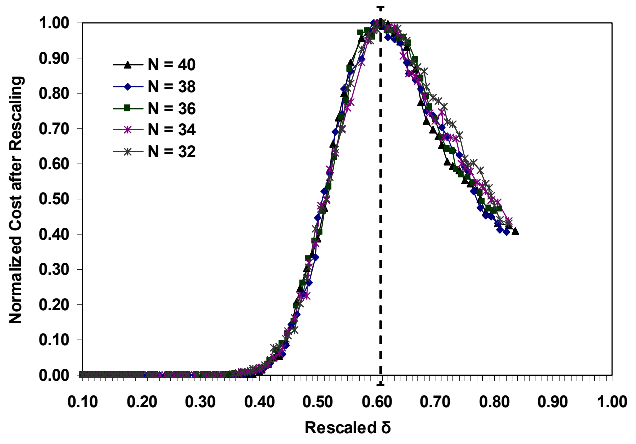

Fig. 6. Computational cost for $p = 1/2$ and various values of n after rescaling.

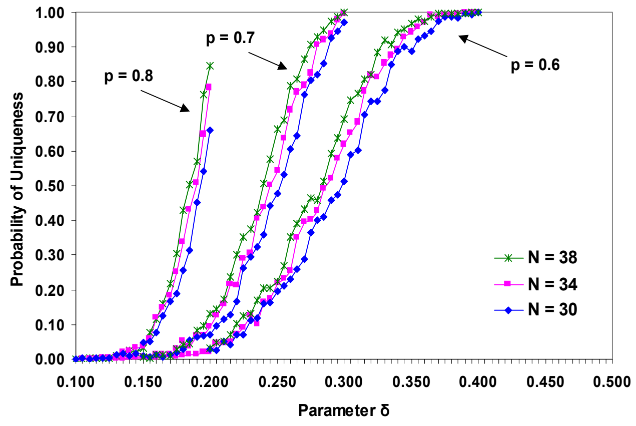

Fig. 7. Phase transition for $p = 0.6, 0.7, 0.8$ and various values of n.

To summarize our findings so far, we have seen that WSAT formulas exhibit phase transitions that are related to the uniqueness probability of the optimal assignment. Furthermore, we saw that by increasing the value of p (Figure 3) one obtains the hardest to solve instances, with easy-hard patterns in their cost. An interesting question is *why* does this happen. We believe that when p is large, most clauses (irrespective of their color) have large weights which makes it extremely difficult to locate the variables that achieve the largest overall weight. When on the other hand p is small, it is the value of δ that defines the hardest instances. Thus in this case we get easy-hard-easy patterns with medium values for δ defining the most difficult to solve problems.

Another interesting question is whether the characterization we obtained using finite size scaling for the case $p = 1/2$ also applies to other values of p. In particular, it is important to know the threshold point for distributions with large values of p as these generate the most interesting formulas from the algorithm designer's point of view.

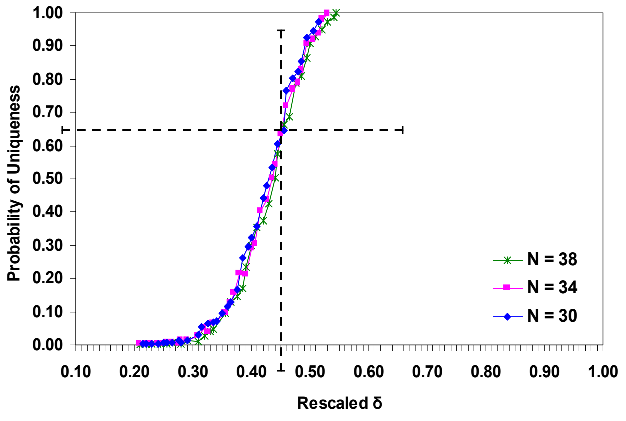

Fig. 8. Phase transition for $p = 0.7$ and various values of n, after rescaling.

Figure 7 shows how the probability of uniqueness changes as a function of δ, when $p = 0.6, 0.7, 0.8$ and $n = 30, 34, 38$. We were able to plot all these curves in the same figure as the separation introduced by increasing the value of p was a lot more than the separation introduced by increasing the value of n. Furthermore, it is worthwhile to observe how the threshold function sharpens up for larger values for p indicating an abrupt change in the uniqueness probability, similar to that observed for the backbone of SAT distributions[11,13–15].

It turns out that the rescaling formula

$$\delta' = \delta n^{\epsilon/2}\sqrt{1-\epsilon} \quad \text{with} \quad \epsilon = 0.56$$

works equally well for other values of p ($\neq \frac{1}{2}$) provided p is kept fixed and only n varies. Figure 8 shows the result of rescaling the curves of Figure 7, when $p = 0.7$. When the same rescaling is applied to the *normalized* computational costs of finding the best assignment for values of $n = 30, 34, 38$, we obtain the universal match shown in Figure 9. The only difference is that the peak value happens for a different value of the rescaled δ (in this case $\delta' = 0.46$). What is remarkable however, is that again the hardest to solve instances *seem* to live at the 65% probability of uniqueness point as shown in Figure 8. Thus using this approach one can concentrate on large values of p (where the really hard WSAT distributions are) and use the rescaling formula to generate the hardest to solve instances.

A final observation is that this methodology cannot be used for values of p very close to 1. Theorem 3 limits the values of p where the hidden assignment is optimal to those where $p \leq 1 - c\frac{\ln n}{n}$. In fact, by some experimentation we were able to determine that the value of the constant c is close to 2. For values of p larger than those implied by the theorem, the generated formulas are essentially the same and no guarantees of optimality can be given.

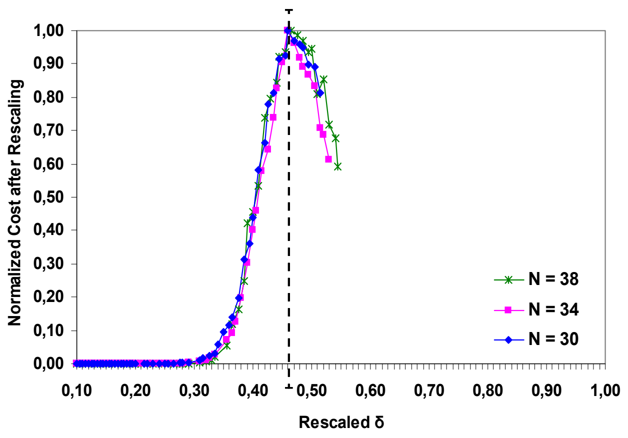

Fig. 9. Computational cost for $p = 0.7$ and various values of n after rescaling.

5 Conclusions

In this work we presented a generator for instances of MAX k-WSAT in which every clause has a weight associated with it and the goal is to maximize the total weight of satisfied clauses. We showed that our generator produces formulas whose hardness can be finely tuned by two parameters p and δ that control the weights of the clauses. Under the right choice of these parameters an easy-hard-easy pattern in the search complexity emerges which is similar to the patterns observed for traditional SAT distributions.

Furthermore, the distributions examined here seem to lie in the middle ground between decision and optimization problems. Increasing the value of p from 0 to 1 has the effect of changing the shape of the computational cost from an easy-hard-easy pattern typical of *decision* problems to an easy-hard pattern typical of *optimization* problems. Thus our distributions seem to bridge the gap between decision and optimization versions of SAT. Furthermore, the hardest instances overall seem to be the ones with the largest value of p.

Finally, we were able to relate these phase transitions with a new structural property of the generated instances which is similar to the backbone of SAT formulas. In particular, we showed how the hardness peak corresponds to a point where there is a transition from formulas which have many optimal assignments to formulas where the optimal assignment is unique. And this is perhaps the most important characteristic of our generator; under the right choice of the parameter δ, not only we know how the optimal solution looks like but we also know it is *unique*. In conclusion, we believe that our generator will be useful in the analysis and development of future SAT heuristics since by knowing what to expect algorithm designers will better test the effectiveness of their search procedures.

Acknowledgements

Many thanks to Christos Papadimitriou for some very useful comments.

References

[1] Noga Alon, Joel H. Spencer. *The probabilistic method*, Wiley, 2000.

[2] Bengt Aspvall, Micahel F. Plass and Robert E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121-123, March 1979.

[3] P. Cheeseman, B. Kanefsky and W.M. Taylor. Where the really hard problems are. In *Proc. IJCAI-91*, pp. 331–337, Australia, 1991.

[4] J. Crawford and L.D. Auton. Experimental results on the cross over point in satisfiability problems. In *Proc. AAAI-93*, pp. 21–27, 1993

[5] P. Erdös and A. Rényi. On the evolution of random graphs. In *Mat Kutato Int. Kozl*, 5, 17–60, 1960.

[6] I. Gent and T. Walsh. The SAT Phase Transition. In *Proc. ECAI-94*, 105-109.

[7] I. Gent and T. Walsh. The TSP Phase Transition. *Artif. Intel.*, 88:349–358, 1996.

[8] T. Hogg, B.A. huberman and C. Williams. Phase transitions and the search problem. *Artif. Intell.*, 81:1-15, 1996.

[9] Kirkpatrick, S. and Selman, B. Critical behavior in the satisfiability of random Boolean expressions. *Science*, 264, 1297-1301, 1994.

[10] Mitchell, D., Selman, B., and Levesque, H.J. Generating hard satisfiability problems. *Artificial Intelligence*, Vol. 81(1-2), 1996. A previous version appeared in *Proc. AAAI-92*, pp. 459–465, San Jose, CA, 1992.

[11] Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., and Troyansky, L. Determining computational complexity from characteristic 'phase transitions'. In *Nature*, Vol. 400(8), 1999.

[12] B. Selman, H. A. Kautz and B. Cohen. Local search strategies for satisfiability testing. In *Second DIMACS Challenge on Cliques, Coloring and Satisfiability*, 1993.

[13] J. Slaney and T. Walsh. Backbones in Optimization and Approximation. In *Proc. IJCAI-01*, 2001.

[14] W. Zhang. Phase transitions and backbones of 3-SAT and MAX 3-SAT. In *Proc. CP-2001.*

[15] W. Zhang and R. E. Korf. A study of complexity transitions on the asymmetric Travelling Salesman Problem. *Artificial Intelligence*, 81:223–239, 1996.