Generating Satisfiable Problem Instances

Dimitris Achlioptas
Microsoft Research
Redmond, WA 98052
optas@microsoft.com

Carla Gomes
Dept. of Comp. Sci.
Cornell Univ.
Ithaca, NY 14853
gomes@cs.cornell.edu

Henry Kautz
AT&T Research
Florham Park, NJ
kautz@research.att.com

Bart Selman
Dept. of Comp. Sci.
Cornell Univ.
Ithaca, NY 14853
selman@cs.cornell.edu

Abstract

A major difficulty in evaluating incomplete local search style algorithms for constraint satisfaction problems is the need for a source of hard problem instances that are guaranteed to be satisfiable. A standard approach to evaluate incomplete search methods has been to use a general problem generator and a complete search method to filter out the unsatisfiable instances. Unfortunately, this approach cannot be used to create problem instances that are beyond the reach of complete search methods. So far, it has proven to be surprisingly difficult to develop a direct generator for satisfiable instances only. In this paper, we propose a generator that only outputs satisfiable problem instances. We also show how one can finely control the hardness of the satisfiable instances by establishing a connection between problem hardness and a new kind of phase transition phenomenon in the space of problem instances. Finally, we use our problem distribution to show the easy-hard-easy pattern in search complexity for *local search* procedures, analogous to the previously reported pattern for complete search methods.

Introduction

In recent years, we have seen the rapid development of both complete and incomplete search methods for constraint satisfaction (CSP) and Boolean satisfiability (SAT) problems. These methods are now applied successfully in a range of applications within artificial intelligence and computer science in general. An important factor in the development of new search methods is the availability of good sets of benchmark problems to evaluate and fine-tune the algorithms. There are two main sources of benchmark problems. One class of benchmarks is based on real-world applications and the other is from random instance generators. Real-world instances are arguably the best source, but unfortunately are often in short supply. Moreover, there is a risk that algorithms are being tuned towards specific application domains for which good benchmarks are available. Random problem generators therefore provide a good additional source

of problem instances. These generators also have the advantage of a more direct control over the problem characteristics, such as size and expected hardness. Hard random instances have led to the development of new stochastic search methods such as Walksat (Selman *et al.* 1996) and the breakout procedure (Morris 1993), and have been used in detailed comparisons of local search methods for graph coloring and related graph problems (Johnson *et al.* 1989). The results of various competitions for CSP and SAT algorithms show that there is a fairly direct correlation between the performance on real-world benchmarks and on hard random instances (DIMACS 1993, 1996; Beijing, 1996; Johnson *et al.* 1989). It is important to note that randomly generated problem instances are not necessarily *unstructured*. Structure may be introduced by translating random problems from one domain into another, or by considering problem domains that by definition exhibit regular structure (Gomes and Selman 1997, Walsh 1999).

Current problem generators are based on recent developments in our understanding of the nature of computationally hard problem instances. In particular, a clear connection has been established between so-called phase transition phenomena and the computational hardness of NP-complete problems (Cheeseman *et al.* 1991, Mitchell *et al.* 1992, Hogg *et al.* 1996). Phase transition phenomena capture the surprisingly sharp transitions from the solvable to the unsolvable in the space of problem instances, as a function of certain problem parameters such as the ratio of the number of constraints to the number of variables. In random distributed problem instances, at low ratios (relatively few constraints) one encounters mostly satisfiable instances, while at high ratios most instances are unsatisfiable. In terms of complexity, one observes a easy-hard-easy pattern, where assignments are easily found in the sat-phase, while inconsistency is easily shown in the unsat-phase. At the phase transition, where roughly half the instances are satisfiable and half the instances are unsatisfiable, one finds a concentration of computationally hard problem instances. The ability to varying the hardness of the problem instances makes it possible to study precisely how different search algorithms *scale* in terms of problem difficulty.

A key limitation of current problem generators concerns their use in the evaluation of incomplete local search methods. This is because the generators generally produce a mixture of solvable (satisfiable) and unsolvable (unsatisfiable)

instances. When a local search style method does not find a solution, it can be difficult to determine whether this is because the algorithm fails to find a solution or because the instance itself is unsolvable. The standard way of dealing with this problem is to use a complete search method to filter out the unsatisfiable cases. However, this limits the size and difficulty of problems instances that can be considered. Ideally, one would use problem generators that generate satisfiable instances only. However, developing such generators has been surprisingly difficult.

As an example, let us consider generating hard satisfiable 3CNF formulas. In order to obtain satisfiable instances only, it is natural to use a strategy where one creates formulas in the phase transition region (ratio of clauses to variables of around 4.25) that are "forced" to have at least one satisfying assignment. To do so, consider the following strategy: generate a random truth assignment T, and then generate a formula with N variables and $4.25N$ random clauses, where one rejects any clause that violates T. This method will in principle generate all possible satisfiable formulas with a clause-to-variable ratio of 4.25 that have T among their solution. What is somewhat surprising however is that the sampling of these formulas is far from uniform: the generator is highly biased towards formulas with many assignments, clustered around T. When fed to local search methods such as Walksat, these formulas are much easier than formulas of comparable size obtained by filtering satisfiable instances from a 3SAT generator. More sophisticated versions of the forced-formula scheme (Asahiro *et al.* 1993, Van Gelder 1993) provide improvements but also lead to biased samples.

There are also a number of theoretical results that show that is is difficult to "hide" a combinatorial object in a larger combinatorial structure. For example, it can be shown that one can easily find cliques over a certain size that are hidden in a random graph, and similar results are known for hiding Hamiltonian cycles (Frieze and McDiarmid 1997). The problem of hiding information in larger combinatorial structures is of interest to the computer science theory community since successful techniques for doing so may eventually lead to more effective cryptographic methods.

Cryptographic problems do suggest one way of creating hard satisfiable problem instances (Impagliazzo *et al.* 1989). For example, Crawford and Kearns (1993) created SAT encodings of the "noisy" parity problem. The instances are guaranteed to have a satisfying assignment but are extremely hard to solve using current SAT procedures. In recent work Massacci (1999) also provides a way of translating the DES crypto protocol into a SAT instance. One can obtain very hard satisfiable instances this way. Since the best algorithms known for dealing directly with the original crypto problem involve exhaustive search, one finds that the best SAT methods are also reduced to an essentially exhaustive search of the space of truth assignments. This means that in practice these problems are in a sense too hard for the development and evaluation of SAT procedures. Furthermore, the cryptographic encodings do not provide a fine-grained way to vary problem hardness in order to studying how the algorithms scale. In general, it seems reasonable to assume that in practical applications one does not expect to find hidden crypto problems, unless one is dealing specifically with a cryptographic application.

In this paper, we will introduce a method for the generation of (empirically) hard satisfiable problem instances. We also show how one can finely control the hardness of the satisfiable instances by establishing a connection between problem hardness and a new kind of phase transition phenomenon in the space of problem instances. As we discussed above, traditional phase transition phenomena involve a sudden transition from a satisfiable to an unsatisfiable phase of the problem instance space. Since our generator only outputs satisfiable instances, such a transition does not occur. However, under the right parameterization, we also observe an easy-hard-easy pattern in the space of satisfiable instances, just as is the case for complete search methods. (For related work, see Clark *et al.* (1996).) This makes it possible to tune the generator to output hard problem instances.

We can link the hardness area to a phase transition which corresponds to a clear threshold phenomenon in the size of the "backbone" of the problem instances. Informally speaking, the backbone measures the amount of shared structure among the set of all solutions to a given problem instance. The size of the backbone is measured in terms of the percentage of variables that have the same value in all possible solutions. We will observe a transition from a phase where the size of the backbone is almost 100% to a phase with a backbone of size close to 0%. The transition is sudden and we will show how it coincides with the hardest problem instances both for incomplete and complete search methods.

Quasigroups with holes

Most traditional benchmark problems are based on randomly generated instances with little or no global structure. In Gomes and Selman (1997), we introduced the so-called quasigroup completion problem in order to obtain benchmark instances with more interesting structural properties.

The best way to view the quasigroup completion problem is in terms of the completion of a Latin square (which technically defines the multiplication table of the quasigroup). Given N colors, a Latin square is defined by an N by N table, where each entry has a color and where there are no repeated colors in any row or column. N is called the *order* of the square. Gomes and Selman considered the problem of whether a partially colored Latin square can be completed into a full Latin square by assigning colors to the open entries of the table. This problem is referred to as the quasigroup completion problem (QCP). QCP is NP-complete (Colbourn 1984) and has an interesting phase transition phenomenon with an associated easy-hard-easy pattern as a function of the fraction of number of preassigned colors. The domain has been used to study the effectiveness of a variety of local consistency measures for constraint satisfaction procedures (Stergiou and Walsh 1999, Walsh 1999, Regin 1994).

The quasigroup completion task has interesting global structure but does not lend itself well for the evaluation of local search methods because we again have a mix of satisfiable and unsatisfiable instances. However, we will introduce a new generator based on the quasigroup domain that gives a natural unbiased way for obtaining only satisfiable

instances, with good computational properties, namely by starting with a full quasigroup and "punching" holes into it. We use a recent result on generating uniformly distributed random complete quasigroups for generating our initial full quasigroup.

The problem of generating uniformly distributed Latin squares is non-trivial. Jacobson and Matthews (1996) show how by simulating an ergodic Markov chain whose stationary distribution is uniform over the space of N by N Latin squares, one can obtain squares that are (approximately) uniformly distributed. The Markov chain Monte Carlo method starts with a complete Latin square. (There is an efficient method for generating a fixed Latin square of any size.) Subsequently, the method randomly "perturbs" the initial Latin square to obtain a new square; repeated random perturbations lead us through a chain of squares. The difficult part is to design sequences of perturbations that lead from one valid Latin square to another while ensuring that one can reach any arbitrary Latin square in the chain with equal probability in the stationary distribution. The method proposed by Jacobson and Matthews corresponds to a random walk on a finite, connected, nonbipartite undirected graph and therefore it is ergodic, with stationary distribution assigning each vertex a probability proportional to its degree.

The Jacobson and Matthews approach provides us with a good starting point for obtaining interesting satisfiable computational instances. We propose the following generator: (1) Generate a complete Latin square according to the Markov chain Monte Carlo approach proposed by Jacobson and Matthews; (2) punch a fraction p of "holes" in the Latin square (*i.e.*, uncolor some of the entries) in a uniformly distributed manner. The resulting partial Latin square is now guaranteed to be satisfiable and moreover, as we will see below, we can finely control its expected hardness by tuning the value of p. We call this new problem the "quasigroup with holes" (QWH) problem.[1] As we will describe below, the instances can be solved directly (in order to test *e.g.,* a constraint-logic programming algorithm) or translated into a Boolean CNF encoding (in order to test general SAT solvers). It is interesting to note that while the quasigroup domain lends itself naturally to a satisfiable instance generator with good computational properties, it is not clear how a similar generator could be developed for, *e.g.*, k-SAT or graph coloring.

The quasigroup with holes problem is NP-hard. This follows from the following argument. Assume one had a polynomial algorithm that could solve QWH. Such an algorithm could be used to solve the quasigroup completion task (QCP), by simply running the algorithm with a polynomial time bound. The bounded algorithm would either solve our completion problem or terminate at the time bound, indicating no solution exists. However, this is impossible because, as noted above, QCP is NP-complete.

In the next sections, we will identify a clear easy-hard-easy pattern for both complete and incomplete search methods on these problem instances. Note that because we are dealing with a distribution of satisfiable instances only, we

---

[1]We thank Mark Stickel for some preliminary discussions on the use of the quasigroups with holes (Stickel, personal communications, May 1998).

obtain a clear full easy-hard-easy diagram for a incomplete search method. Clark *et al.* (1996) provide initial results on a such a pattern for local search using standard benchmarks. However, given the rareness of satisfiable instances on the unsat side of the phase transition it is difficult to establish a clear full pattern. We will also show that the hardness region of our satisfiable problem instances coincides with a new kind of phase transition. This transition differs from the standard sat/unsat transition because we now have only satisfiable instances, but like the standard transition, it is based on an underlying structural property — namely, the backbone.

Problem hardness

In order to solve QWH instances, we explored a range of algorithms. We used an ILOG constraint solver working directly on the constraint satisfaction encoding of the problem. In the ILOG solver, we incorporated, aside from the standard constraint propagation methods, the all-diff constraint (Stergiou and Walsh 1999; Regin 1994). We also implemented (in C) a local search procedure working directly on the constraint representation. Finally, we converted the problem instances into Boolean satisfiability encodings and used state-of-the-art SAT solvers, both complete and incomplete methods. To our surprise, the approach via a SAT encoding is more efficient than using the direct CSP approaches; apparently, the increase in the size of the encoding when going to SAT does not hurt overall performance. Given the space limitations of this paper, we will only include the data for our best performing procedures, the backtracking SAT solver Satz (Li and Anbulagan 1997) and the local search SAT solver Walksat (Selman *et al.* 1996). (Both solvers are available from SATLIB (Hoos 1999).) Our data for the CSP approach is qualitatively the same. The QWH instances thus provide a good benchmark for both CSP and for SAT methods. Experimental data, instances, and generator (both SAT and CSP representation) are available from the authors.

In Fig. 1, we show the computational cost profiles for an incomplete (Walksat; left panel) and a complete (Satz; right panel) search method for the QWH problem. Along the horizontal axis, we vary the fraction of holes in the quasigroup. More specifically, we take the ratio of the number of holes to the total number of entries in the Latin square, *i.e.*, N^2, where N is order of the square. The vertical axis gives the median computational cost. For Walksat, the cost is measured in terms of the total number of variable flips; for Satz we measured the total number of backtracks.

The figure shows a clear easy-hard-easy pattern for both the incomplete and the complete search methods. Over a range of different sizes ($N = 30, 33, 36$) we see a rapid (in fact, exponential) increase in search cost in the hardest region. Close observation shows that there is a slight shift in the location of the peaks. We will return to this issue below, when we discuss a way of rescaling the figures to precisely line up the peaks.

Aside from having a clear easy-hard-easy pattern, the main point of interest in Fig. 1 is the profile for the incomplete search method. We see a clear example of an easy-hard-easy pattern for an incomplete search method. Because previous problem generators give a mixture of sat and un-
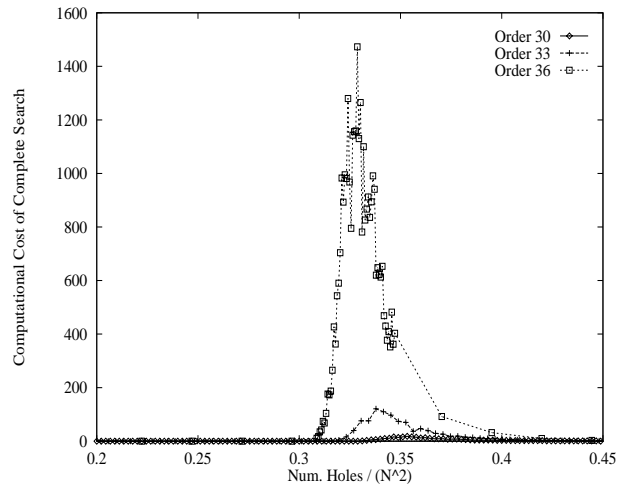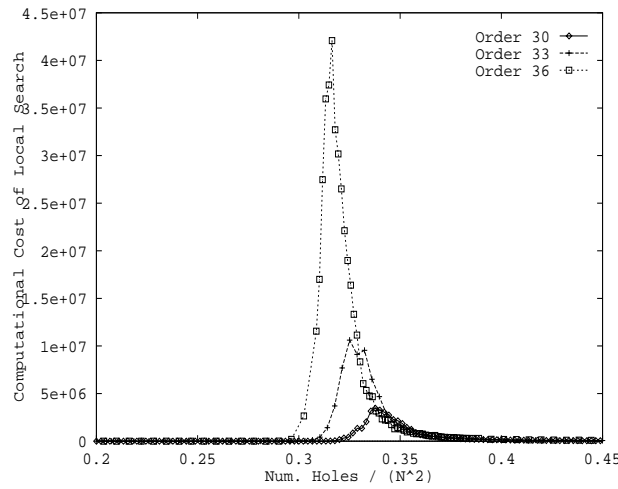
Figure 1: Computational cost profiles for incomplete (Walksat) and complete (Satz) search methods for QWH.

sat cases, such an easy-hard-easy pattern has generally been reported so far only for complete methods, which can handle both types of instances. Our figure shows that the notions of under-constrained, critically constrained, and over-constrained (Hogg *et al.* 1996) are also predictive of the performance of incomplete search methods.
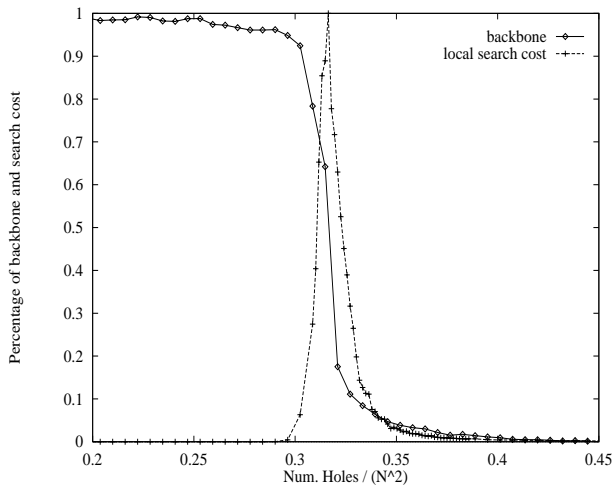

Figure 2: Backbone phase transition with cost profile.

A New Type of Phase Transition

One of the key advances in our understanding of problem hardness has been the connection between the easy-hard-easy pattern in search complexity and phase transition phenomena (Cheeseman 1991; Mitchell *et al.* 1992; Kirkpatrick and Selman 1994; Hogg *et al.* 1996; Hayes 1996). In particular, a clear connection has been established between the hardest problem instances and the phase transition region, where instances shift from being mostly satisfiable to being mostly unsatisfiable. One of the interesting aspects of this connection is that properties of the SAT/UNSAT phase transition can be analyzed quite independently from any particu-
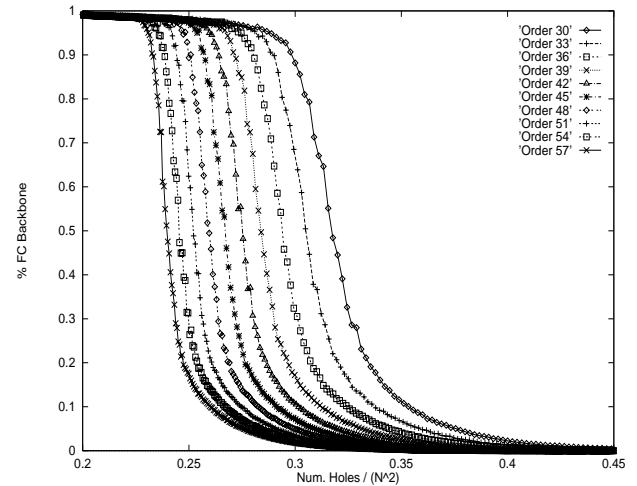

Figure 3: Backbone for different orders.

lar solution procedure. In fact, this has led to a large number of papers on the SAT/UNSAT phase transition *per se*.

For the QWH instances, we do not have a SAT/UNSAT phase transition, since all our instances are guaranteed to be satisfiable. Nevertheless, we can use recently introduced notions from the study of phase transition phenomena to link the peak in search complexity to a phase transition in structural properties of our problem instances. To do so, we will consider so-called backbone variables.

Monasson *et al.* (1999) introduced the notion of the *backbone* of a SAT problem to refer to the fraction of its variables that are *fully constrained*: that is, which take on the same values in all solutions. The backbone fraction (ratio of backbone variables to the total number of variables) is a property of CSP and SAT problems that is well-defined for satisfiable distributions.

Fig. 2 shows the backbone fraction as a function of the fraction of holes in the QWH problem. We also included the normalized cost of local search. The figure shows a
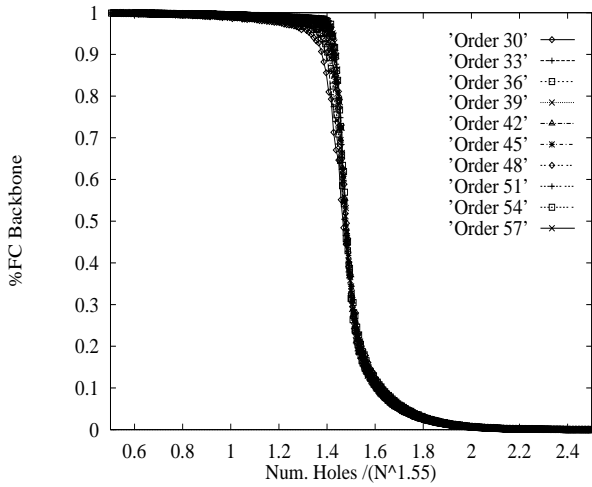
Figure 4: Backbone for different orders (rescaled).

Figure 5: Normalized computational cost.

Figure 6: Re-parameterized computational cost.

sharp phase transition phenomenon in the backbone fraction, which coincide with the hardness peak in local search.[2]

The reasons for the correlation between problem hardness and the appearance of the backbone are not fully understood at this time. One intuition is that backtracking search algorithms have the worst performance when they make an incorrect choice near the root of the search tree: that is, when they make a variable-value assignment that appears in no solution. For the algorithm to have a significant chance of making such a bad choice a non-negligible fraction of the variables must appear in the backbone. When the backbone fraction nears 1, however, the problems are so overconstrained that incorrect choices near the root are quickly detected and corrected. For local search procedures, an explanation might be developed by considering the relationship between the backbone and set of solutions to the instances. When the backbone is small, there are many solutions widely distributed in the search space, and so local search may quickly find one. When the backbone is near 1, the solutions are tightly clustered, so that that all clauses "vote" to push the search in the same direction. A partial backbone, however, may indicate that solutions are in different clusters that are widely distributed, with different clauses pushing the search in different directions. Making these intuitions precise, however, awaits future research.

Re-parameterization

As we noted above, there is a slight shift in the location of the hardness peak as a function of N. There is a similar shift in the location of the backbone phase transition. This points to the fact that the original parameterization in terms of the fraction of holes does not exactly capture the dimensionality of our problem.[3] Fig. 3 shows the shift in the backbone tran-

sition for a larger range of problem sizes ($N = 30, \ldots, 57$).[4]

Some experimentation with different parameterization leads us to Fig. 4. This figure shows the backbone plotted against the number holes over $N^{1.55}$. Note that we originally used "number of holes over N^2". We are currently working on an analytical derivation of the re-parameterization.

Finally, Figs. 5 and 6 show how our rescaling also corrects for the shift in the complexity peak of our local search method. To show the original shift, Fig. 5 gives the search complexity for three different sizes of the QWH problem, where the cost has been normalized to 1. Fig. 6 shows how the peaks collapse onto each other after rescaling. The peaks for the complete search method (right panel in Fig. 1) also

---

[2]The figure gives data for $N = 36$. The hardness peak for our complete search method also lies in the phase transition region but is shifted slightly to the right. We are currently investigating whether that shift is real or part of the uncertainty in our data.

[3]Note that a similar shift is also present in the original quasigroup completion problem.
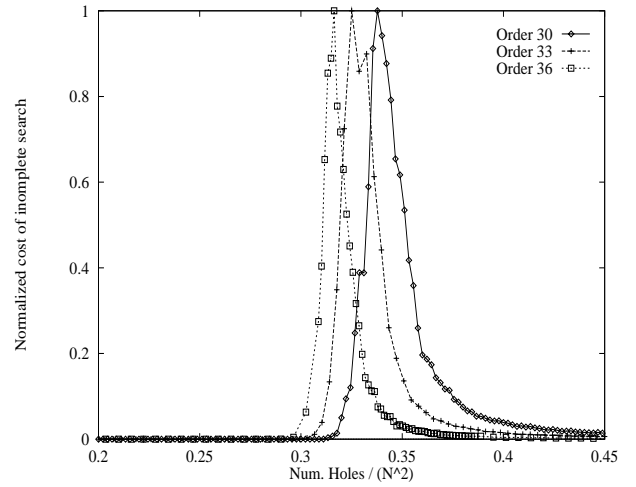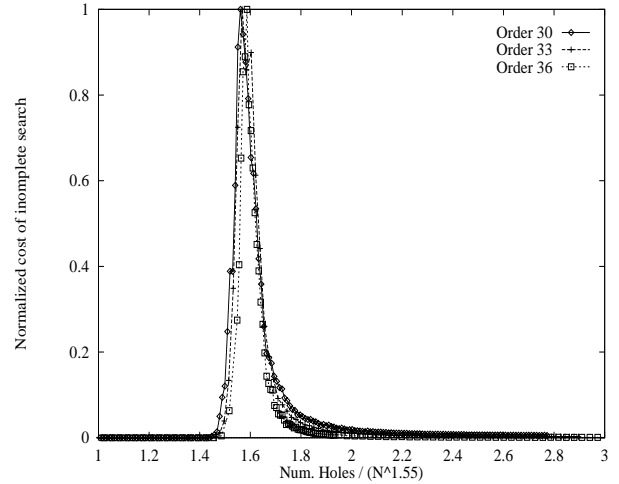
[4]Computing the full backbone is prohibitively expensive. The figure gives a good approximation of the backbone fraction computed by using forward-checking to estimate the fraction of fixed variables. This estimate is a few percentage off from the true value, but the shifting behavior appears identical to that of the full backbone, based on experiments for smaller values of N.

align after such a rescaling.

Conclusions

We propose a problem generator for satisfiable instances. The generator samples from *satisfiable* quasigroups of a given size with a given number of holes. The hardness of the QWH problem instances can be tuned by varying the fraction of holes in the quasigroup instances. The main advantage of this generator is that it generates satisfiable instances only and is therefore well-suited for use in the study and evaluation of incomplete search methods.

Several earlier attempts at designing such a generator (*e.g.,* by forcing a given solution during the problem generation) were unsatisfactory. Using our generator, we showed that a local search method does exhibit the easy-hard-easy pattern, as observed previously for complete search methods. Based on the notion of under-constrained, critically constrained, and over-constrained regions identified with complete search methods, it was believed that an easy-hard-easy pattern would emerge for local search methods but this was difficult to confirm empirically because satisfiable instances in the over-constrained region are extremely rare for standard problem generators.

We also show how the hardest region of the satisfiable instances coincides with a new kind of phase transition in terms of the backbone of the problem instances. The backbone characterizes the amount of shared structure between solutions. Finally, we present an empirically obtained re-parameterization of the phase transition and complexity peak of the quasigroup with holes problem. Our generator outputs instances suitable for both CSP and SAT style methods. The generator should therefore be of use in the future development of stochastic local search style CSP and SAT methods.

References

Asahiro, Y., Iwama, K. and Miyano, E. (1993). Random generation of test instances with controlled attributes. In *DIMACS 1993*, op cite.

Beijing (1996). *International Competition and Symposium on Satisfiability Testing*, Beijing, China, March 15–17, 1996.

Cheeseman, P. and Kanefsky, R. and Taylor, W. (1991). Where the Really Hard Problems Are. *Proc. IJCAI–91,* 1991, 163–169.

Clark, D.A., Frank, J., Gent, I.P., MacIntyre, E., Tomov, N., Walsh, T. (1996). Local search and the number of solutions. *Proc. CP-96*, 1996.

Colbourn, C. (1984). The Complexity of Completing Latin Squares. *Discrete Appl. Math.*, 8, (1984), 25-30.

Crawford, J. and Kearns, M. (1993). Instances for learning the parity function. Unpublished note, see Hoos (1999).

DIMACS (1993). *Second DIMACS Implement. Challenge, 1993*. Pub. as *DIMACS Series in Disc. Math. and Theor. Comp. Sci.*, vol. 26, D. Johnson and M. Trick, eds., AMS, 1996.

DIMACS (1996). *Satisfiability Problem*, DIMACS Workshop, 1996. Pub. as *DIMACS Discrete Math. and Theor. Comp. Sci.*, vol. 35, D. Du, J. Gu, and P. Pardalos, eds., AMS, 1997.

Frieze, A. and McDiarmid, C. (1997). Algorithmic theory of random graphs. *Random Structures and Algorithms*, vol. 10 (1997) 5–42.

Gent, I. and Walsh, T. (1993) An empirical analysis of search in GSAT. *J. of Artificial Intelligence Research*, vol. 1, 1993.

Gomes, C.P. and Selman, B. (1997a). Problem structure in the presence of perturbations. *Proc. AAAI-97*, 1997.

Hayes, B. (1996). Can't get no satisfaction. *American Scientist* vol. 85, 108 (1996)

Hogg, T., Huberman, B.A., and Williams, C.P. (Eds.) (1996). Phase Transitions and Complexity. *Artificial Intelligence*, 81 (Spec. Issue), 1996.

Hoos, H. 1999. SATLIB. A collection of SAT tools and data. See www.informatik.tu-darmstadt.de/AI/SATLIB.

Impagliazzo, R., Levin, L., and Luby, M. (1989). Pseudo-random number generation from one-way functions. *Proc. 21st STOC*, 1989, 12-24.

Jacobson, M.T. and Matthews, P. (1996) Generating uniformly distributed random latin squares. *J. of Combinatorial Designs*, vol. 4., no. 6, (1996) 405–437.

Johnson, D.S. , Aragon, C.R., McGeoch, L.A., and Shevon C. (1989) Optimization by Simulated Annealing: An Experimental Evaluation. *Operations Research*, 37:6 (1989), 865-892.

Kirkpatrick, S. and Selman, B. (1994). Critical behavior in the satisfiability of random Boolean expressions. *Science*, 264, 1994, 1297–1301.

Li, Chu Min and Anbulagan (1997). Heuristics based on unit propagation for satisfiability problems. *Proc. IJCAI-97,* 366–371.

Massacci, F. (1999). Using Walk-SAT and Rel-SAT for cyptographic key search. *Proc. IJCAI-99*, 1999, 290-295.

Mitchell, D. and Levesque H. (1996). Some pitfalls for experimenters with random SAT. *Artificial Intelligence*, Vol. 81(1-2), 1996, 111–125.

Mitchell, D., Selman, B., and Levesque, H.J. (1992). Hard and easy distributions of SAT problems. *Proc. AAAI-92*, San Jose, CA (1992) 459–465.

Morris, P. (1993) The breakout method for escaping from local minima. *Proc. AAAI-93*, 1993, 40–45.

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., and Troyansky, L. (1996). Determining computational complexity from characteristic 'phase transitions'. *Nature*, Vol. 400(8), 1999.

Regin, J.C. (1994). A filtering algorithm for constraints of difference in CSP. *Proc. AAAI-94*, 1994, 362–367.

Selman, B. and Levesque, H.J., and Mitchell, D.G. (1992). A New Method for Solving Hard Satisfiability Problems.

Selman, B., Kautz, H.A., and Cohen, B. (1996). Local search strategies for satisfiability testing. In DIMACS (1993).

Shaw, P., Stergiou, K., and Walsh, T. (1998) Arc consistency and quasigroup completion. *Proc. ECAI-98*, workshop on binary constraints, 1998.

Stergiou, K. and Walsh, T. (1999) The Difference All-Difference Makes Proc. of *IJCAI-99*, Stockholm, Sweden.

Walsh, T. (1999) Search in a Small World. Proc. of *IJCAI-99*, Stockholm, Sweden, 1999.

Van Gelder, A. (1993). Problem generator (mkcnf.c) contributed to the DIMACS 1993 Challenge archive.