Balance and Filtering in Structured Satisfiable Problems

Henry Kautz
Yongshao Ruan
Dept. Comp. Sci. & Engr.
Univ. Washington
Seattle, WA 98195
kautz@cs.washington.edu
ruan@cs.washington.edu

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

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

Mark Stickel
Artificial Intelligence Center
SRI International
Menlo Park, California 94025
stickel@ai.sri.com

Abstract

New methods to generate hard random problem instances have driven progress on algorithms for deduction and constraint satisfaction. Recently Achlioptas *et al.* (AAAI 2000) introduced a new generator based on Latin squares that creates only *satisfiable* problems, and so can be used to accurately test incomplete (one sided) solvers. We investigate how this and other generators are biased away from the uniform distribution of satisfiable problems and show how they can be improved by imposing a *balance* condition. More generally, we show that the generator is one member of a family of related models that generate distributions ranging from ones that are everywhere tractable to ones that exhibit a sharp hardness threshold. We also discuss the critical role of the problem encoding in the performance of both systematic and local search solvers.

1 Introduction

The discovery of methods to generate hard random problem instances has driven progress on algorithms for propositional deduction and satisfiability testing. Gomes & Selman (1997) introduced a generation model based on the quasigroup (or Latin square) completion problem (QCP). The task is to determine if a partially colored square can be completed so that no color is repeated in any row or any column. QCP is an NP-complete problem, and random instances exhibit a peak in problem hardness in the area of the phase transition in the percentage of satisfiable instances generated as the ratio of the number of uncolored cells to the total number of cells is varied. The structure implicit in a QCP problem is similar to that found in real-world domains: indeed, many problems in scheduling and experimental design take the form of a QCP. Thus, QCP complements earlier simpler generation models, such as random k-cnf (Mitchell *et al.* 1992). Like them QCP generates a mix of satisfiable and unsatisfiable instances.

In order to measure the performance of incomplete solvers, it is necessary to have benchmark instances that are known to be satisfiable. This requirement is problematic in domains where incomplete methods can solve larger instances than complete methods: it is not possible to use a complete method to filter out the unsatisfiable instances. It has proven difficult to create generators for satisfiable k-sat. Achlioptas *et al.* (2000) described a generation model for satisfiable quasigroup completion problems called "quasigroups with holes" (QWH). The QWH generation procedure basically inverts the completion task: it *begins* with a randomly-generated completed Latin square, and then erases colors or "pokes holes". The *backbone* of a satisfiable problem is the set of variables that receive the same value in all solutions to that problem. Achlioptas *et al.* (2000) showed that the hardest QWH problems arise in the vicinity of a threshold in the average size of the backbone.

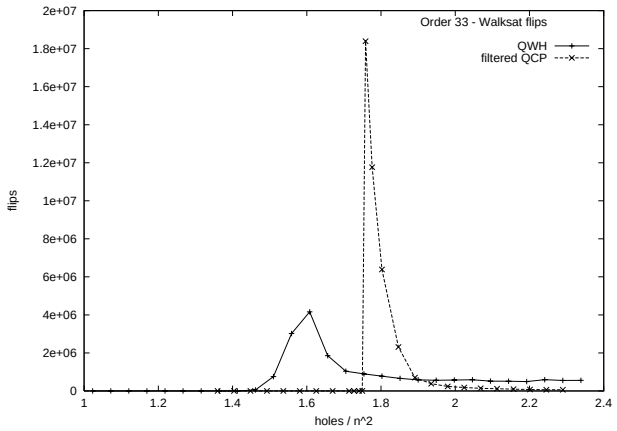

Figure 1: Comparison of the QWH and filtered QCP models for order 33 using Walksat. The x-axis is the percentage of holes (re-parameterized) and the y-axis is the number of flips.

model	Walksat flips	Satz backtracks	Sato branches
QWH	$4e+6$	41	$3e+5$
filtered QCP	$1e+8$	394	$1.5e+6$
balanced QWH	$6e+8$	4,984	$8e+6$

Table 1: Peak median hardness for different generation models for order 33 problems.

Despite the similarities between the filtered QCP and QWH models the problem distributions they generate are quite different. As noted by Achlioptas *et al.* (AAAI 2000), the threshold for QCP is at a higher ratio of holes than is the threshold for QWH. But even more significantly, we discovered that the hardest problems obtained using the filtered QCP

model are about an order of magnitude more difficult to solve than the hardest one obtained using the QWH model. Figure 1 illustrates the difference for the incomplete local search solver Walksat (Selman *et al.* 1992). (The critical parameter for a fixed order n is the percentage of holes (uncolored squares) in the problem. In all of the graphs in this paper the $x-$axis is re-parameterized as the number of holes divided by $n^{1.55}$. Achlioptas *et al.* (2000) demonstrates that this re-parameterization adjusts for different problem sizes.) The first two lines of Table 1 compare the peak hardness of QWH and filtered QCP for Walksat and two complete systematic solvers, Satz (Li & Anbulagan 1997) and Sato (Zhang 1997). These solvers are running on Boolean CNF encodings of the problem instances. The performance of such modern SAT solvers on these problems is competitive with or superior to the performance of optimized CSP solvers running on the direct CSP encoding of the instances (Achlioptas *et al.* 2000). (Note that the number of backtracks performed by Satz is usually much lower than the number performed by other systematic algorithms because of its use of lookahead.)

We begin by explaining this difference in hardness by showing how each generation model is *biased* away from the uniform distribution of all satisfiable quasigroup completion problems. Next, we introduce a new satisfiable problem model, *balanced QWH*, which creates problems that are even harder than those found by filtered QCP, as summarized in the last line of Table 1. This new model provides a tool for creating hard structured instances for testing incomplete solvers. In addition, its simplicity and elegance suggests that it will provide a good model to use for theoretical analysis of structured problems.

In the final section of the paper we turn to the issue of how we encode the quasigroup problems as Boolean satisfiability problems. For systematic methods the best encoding is based on a view of a quasigroup as a 3-dimensional cube, rather than as a 2-dimensional object. As shown below, the 2-D encodings are almost impossible to solve by systematic methods for larger orders. This difference is not unexpected, in that the 3-D encodings include more redundant constraints, which are known to help backtracking algorithms. For Walksat, however, a more complex picture emerges: the hardest instances can be solved more quickly under the 2-D encoding, while under-constrained instances are easier under the 3-D encoding.

2 The QWH and Filtered QCP Models

In order to better understand the QWH and filtered QCP models we begin by considering the problem of choosing a member uniformly at random from the set of all satisfiable quasigroup completion problems of order n with h uncolored cells. It is easy to define a generator for the uniform model: (1) Color $n^2 - h$ cells of an order n square randomly; (2) Apply a complete solver to test if the square can be completed; if not, return to step (1). This generator for the uniform model is, however, impractical for all but largest values of h. The problem is that the probability of creating a satisfiable instance in step (1) is vanishingly small, and so the generator will loop for an exponentially long time. In other words, the set of satisfiable quasigroup completion problems, although large, is small relative to the set of all completion problems.

Because the instances generated by placing down random colors are almost certainly unsatisfiable, the quasigroup completion generator from Gomes & Selman (1997) tries to filter out as many unsatisfiable configurations while incrementally partially coloring a square. The formulation is in terms of a CSP, with a variable for each cell, where the domain of the variable is initialized to the set of n colors. The generator repeatedly selects an uncolored cell at random, and assigns it a value from its domain. Then, forward-checking is performed to reduce the domains of cells that share a row or column with it. If forward-checking reaches an inconsistency (an empty domain) the entire square is erased and the process begins anew. Otherwise, once a sufficient number of cells have been colored arc consistency is checked; if the square passes this test, then it is output. The hardest mix of sat and unsat instances is generated when the number of holes is such that about 50% of the resulting squares are satisfiable. Because of it's popularity in the literature, we simply call the distribution generated by the process the QCP model. Adding a final complete solver to check satisfiability and eliminating those square which are unsatisfiable results in the filtered QCP model.[1]

The hardest hole ratio for the filtered QCP model is shifted toward the more constrained side. We discovered that hardest satisfiable instances occur when the filtering step eliminates about 90% of the partial Latin squares.

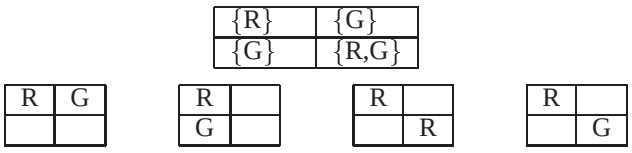

Figure 2: Top: the domains of each variable of an order 4 Latin square after the top left cell is colored R. Bottom: the four completion problems, generated with probabilities $1/3, 1/3, 1/6, 1/6$ respectively.

While the use of incremental forward checking successfully biases QCP away from unsatisfiable problems, it also has the unintended effect of introducing a bias between different *satisfiable* instances. For a simple example, consider generating order 2 problems with 2 holes, where the colors are R and G. Without loss of generality, suppose the first step of the algorithm is to color the top left square R. As shown in Figure 2, forward-checking reduces the domains of two of the remaining cells. The next step is to pick one of the three remaining cells at random, and then choose a color in its domain at random. As a result, one of the four partial Latin squares shown in the figure is created, but with different probabilities: the first two with probability $1/3$ each, and the last two with probability $1/6$ each. The fourth square is immediately eliminated by the final forward checking step. Because our choice of initial square and color was arbitrary, we see that each of the partial Latin squares where the two colored cells are in the same row or column is twice as like to be generated as one in which the two colored cells appear on a diagonal.

The QWH model introduced by Achlioptas *et al.* (2000) works as follows: (1) A random complete Latin square is cre-

[1]The QCP generator used in Gomes & Selman (1997) did not perform the final arc-consistency test. Thus, it generated a higher percentage of unsatisfiable instances. The filtered (satisfiable) distribution from that generator and ours are identical.

Figure 3: Two partial Latin squares with same pattern of holes but a different number of solutions, illustrating the bias of the QWH model.

ated by a Markov-chain process; (2) h random cells are uncolored. Does this generate the uniform model? Again the answer is no. Because all patterns of h holes are equally likely to be generated in step (2), we can without loss of generality consider both the number and pattern of holes to be fixed. The probability of creating a particular completion problem is proportional to the number of different complete Latin squares that yield that problem under the fixed pattern of holes. Therefore, we can show that QWH does not generate the uniform model by simply presenting two partial Latin squares with the same pattern of holes but a different number of solutions.

Such a counterexample appears in Figure 3: The square on the left has two solutions while the one on the right has only a single solution. In other words, the left square is twice as likely to be generated as the right square. In general, the QWH model is biased towards problems that have many solutions.

3 Patterns and Problem Hardness

We now consider different models of QCP and QWH, corresponding to different patterns. As we will see, the complexity of the models is dependent not only on the number of uncolored cells but also on the underlying pattern.

3.1 Rectangular and Aligned Models

In order to establish a baseline, we first consider two tractable models for QCP and QWH, which we refer to as *rectangular* and *aligned* models. Figure 4 (left and middle) illustrates such instances. In the *rectangular* QWH model, a set of columns (or rows) is selected and all the cells in these columns are uncolored. Moreover, one additional column (row) can be selected and be partially uncolored. The *aligned QWH* model is a generalization of the rectangular model in that we can pick both a set of rows and a set of columns and treat them as the chosen rows/columns in the rectangular model. Put differently, in the aligned model, the rows and columns can be permuted so that all cells in $C = \{1, \ldots, r-1\} \times \{1, \ldots, s-1\}$ are colored, some cells of $\{1, \ldots, r\} \times \{1, \ldots, s\} \setminus C$ are colored and all other cells are uncolored. We note that one could also naturally generate rectangular and aligned QCP instances.

In order to show that both the rectangular and aligned models are tractable, let us consider a Latin rectangle R on symbols $1, \cdots, n$. Let R(i) denote the number of occurrences of the symbol i in R, $1 \leq i \leq n$. We first introduce the following theorem from combinatorics.

Theorem: (Ryser 1951) An r x s Latin rectangle R on symbols $1, \cdots, n$ can be embedded in a Latin square of side n if and only if

$$R(i) \geq r + s - n \text{ for all } i, 1 \leq i \leq n.$$

Theorem: Completing a rectangular QCP or QWH is in P.

Proof: We can complete the rectangular QCP or QWH instance column by column (or row by row), starting with the column (or row) partially uncolored. (If there is no partially uncolored columns or rows, consider an arbitrary fully uncolored column or row.) We construct a bipartite graph G, with parts $U = \{u_1, u_2, \ldots, u_m\}$, and $W = \{w_1, w_2, \ldots, w_m\}$ in the following way: U represents the uncolored cells of the column (row) that needs to be colors; W represents the colors not used in that column (row). An edge (u_i,w_j) denotes that node u_i can be colored with color w_j. To complete the first column (row) we find a perfect matching in G. If there is no such matching we know that the instance is unsatisfiable. After completing the first column (row), we can complete the remaining columns (rows) in the same way.QED

Theorem: Completing an aligned QCP or QWH is in P.

Proof: We start by noting that an aligned instance can be trivially rearranged into an r x s rectangle, $s \leq n$, by permutating rows and columns (with possibly a row or column partially uncolored). So, if we show that we can complete an r x s rectangle into an r x n rectangle, we have proved our theorem, since we obtain an instance of the rectangular model. To complete an r x s rectangle into an r x n, again we complete a column (or row) at a time, until we obtain a rectangle of side n. For the completion of each column (or row), again, we solve a matching on the bipartite graph. The only difference resides in the way we build the graph: W only contains colors for which the condition in Ryser's theorem is satisfied. QED

3.2 Balance

While the rectangular and aligned models cluster holes, we now turn to a model that attempts to increase problem hardness by minimizing clustering. Let us start by reviewing the role of balance in the two most studied combinatorial optimization problems over random structures: random satisfiability and random graph coloring. It is particularly interesting to note the features shared by algorithms performing well on these problems.

Random k-SAT. A random formula is formed by selecting uniformly and independently m clauses from the set of all $2^k \binom{n}{k}$ k-clauses on a given set of n variables. The first algorithm to be analyzed on random k-SAT employs the *pure literal* heuristic repeatedly: a literal ℓ is satisfied only if $\bar{\ell}$ does not appear in the formula. Thus, a pure variable has all its occurrences appear with the same sign – a rather dramatic form of sign imbalance. The next key idea is unit-clause propagation, *i.e.*, immediately satisfying all clauses of length 1. More generally, dealing with shortest clauses first has turned out to be very useful. Subsequent improvements also come from exploiting imbalances in the formula: using degree information to determine which variable to set next and considering the number of positive and negative occurrences to determine value assignment.

Bayardo & Schrag (1996) gave experimental results on the role of balance, by considering random k-SAT formulas in which all literals occur in the same number of clauses. In such formulas, an algorithm has to first set a non-trivial fraction of all variables (essentially blindly) before any of the ideas mentioned above can start being of use. This suggests the potential of performing many more backtracks and indeed,

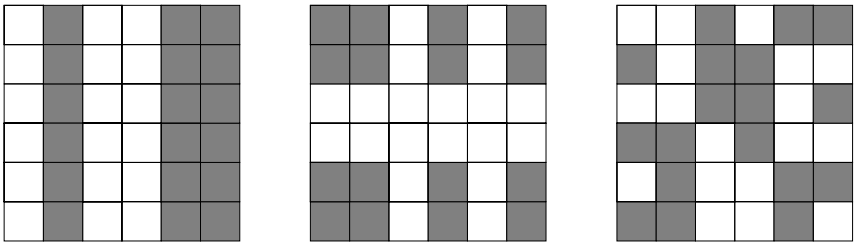

Figure 4: Left: an example of the rectangular model; middle: an example of the aligned model; right: a balanced model. Holes are in white.

balanced random k-SAT formulas are an order of magnitude harder than standard ones.

Random graph coloring. A random graph on n vertices is formed by including each of the $\binom{n}{2}$ edges with probability p. For graph coloring the most interesting range is $p = d/n$, where d is a constant. One can also consider *list-coloring*, where each vertex has a prescribed list of available colors. Analogously to the pure literal heuristic, a first idea is to exploit the advantage offered by low degree vertices. In particular, if we are using k colors then we can safely remove from the graph all vertices having degree smaller than k: if we can color the remaining graph then we can certainly complete the coloring on these vertices since each one of them will have at least one available color (as it has at most $k-1$ neighbors). Upon potentially reaching a k-core, where every vertex has degree at least k, it becomes useful to consider vertices having fewest available colors remaining and, among them, those of highest degree. Conversely, random graphs that are degree-regular and with all lists having the same size tend to be harder to color.

3.3 Balance in Random Quasigroups

In the standard QWH model we pick a random quasigroup and then randomly turn a number of its entries to holes. Fixing the quasigroup choice and the number of holes, let us consider the effect that different hole-patterns have on the hardness of the resulting completion problem. (For brevity, we only refer to rows below but naturally all our comments apply to columns just as well.)

Two extreme cases are rows with just one hole and rows with n holes. In the first case, the row can be immediately completed, while in the second case it turns out that given any consistent completion of the remaining $n \times (n-1)$ squares one can always complete the quasigroup. More generally, it seems like a good idea for any algorithm to attempt to complete rows having a smallest number of holes first, thus minimizing the branching factor in the search tree. Equivalently, having an equal number of holes in each row and column should tend to make things harder for algorithms.

Even upon deciding to have an equal number of holes in each row and column, the exact placement of the holes remains highly influential. Consider for example a pair of rows having holes only in columns i, j and further assume that there are no other holes in columns i, j. Then, by permuting rows and columns it is clear that we can move these four holes to, say, the top left corner of the matrix. Note now that in this new (isomorphic) problem the choices we make for these four holes are independent of all other choices we make in solving the problem, giving us a natural partition to two independent subproblems.

More generally, given the 0/1 matrix A where zeros correspond to colored entries and ones correspond to holes, one can attempt to permute the rows and columns to minimize the *bandwidth* of A (the maximum absolute difference between i and j for which $A(i,j)$ is 1). Having done so, the completion problem can be solved in time exponential in the bandwidth.

We were surprised to discover that even though Satz contains no code that explicitly computes or makes use of bandwidth (indeed, *exactly* computing the bandwidth of a problem is NP-complete), it is extremely efficient for bounded-bandwidth problems. We generated bounded-bandwidth instances by first punching holes in a band along the diagonal of a Latin square, and then shuffling the rows and columns 1,000 times to hide the band. Satz solved all instances of this type for all problem sizes (up to the largest tested, order 33) and hole ratios in either 0 or 1 backtrack! Satz's strategy is Davis-Putnam augmented with one-step lookahead: this combination is sufficient to uncover limited-bandwidth instances.

When the holes are placed randomly then with high probability the resulting matrix A will have high bandwidth. Alternatively, we can view A as a random $n \times n$ bipartite graph in which vertex i is connected to vertex j iff there is a hole in position (i, j). The high bandwidth of A then follows from relatively standard results from random graph theory (Fernandez de la Véga 1981). Moreover, having an equal number of holes in each row/column makes the random graph regular, which *guarantees* that A has large bandwidth. Intuitively, small balanced instances should exhibit properties that hold of large random instances in the asymptotic limit.

Viewing the hole pattern as a regular random bipartite graph readily suggests a way for generating a uniformly random hole pattern with precisely q holes in each row and column for any q (*i.e.*, $q = h/n$) by the following algorithm:

Set $H = \emptyset$.
Repeat q times:
  Set T to $\{1, \ldots, n\} \times \{1, \ldots, n\} \setminus H$.
  Repeat n times:
    Pick a uniformly random $(i, j) \in T$;
    Add (i, j) to H;
    Remove all elements in row i and column j from T.

Balancing can be applied to either the QWH or QCP models: in the former, the pattern is used to uncolor cells; in the latter, the pattern is used to determine the cells that are *not* colored incrementally.

3.4 Empirical Results

We measured the difficulty of solving problem distributions generated under each of the models using three different al-

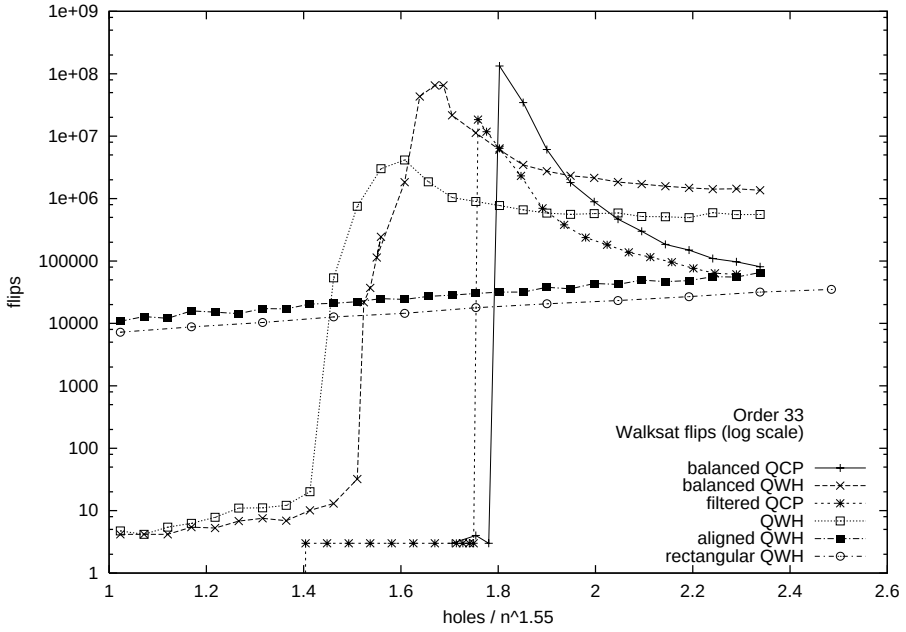

Figure 5: Comparison of generation models for order 33 using Walksat. Y-axis is the number of flips (log scale) and x-axis is the percentage of holes (re-parameterized). Each data point represents the median number of flips for 100 instances. Less-constrained instances appear to the right.

gorithms. As a preliminary step problems were simplified by arc consistency and then translated into SAT problems in conjunctive normal form, as described in Sec. 4 below. For each data point for a range of percentage of holes 100 problems were generated and solved by three different algorithms: Walksat, which performs local search over the space of truth assignments, using 30% noise and no cutoff; Satz, which implements the backtracking Davis-Putnam algorithm augmented with 1-step lookahead; and Sato (Zhang 1997), another Davis-Putnam type solver that uses dependency-directed backtracking and clause learning. In this paper we present the data for order 33, a size which clearly distinguishes problem difficulty but still allows sufficient data to be gathered. For reasons of space, we will omit detailed results for Sato, which are qualitatively similar to those for Satz.

Figure 5 displays the results for Walksat: note the log scale, so that each major division indicates an *order of magnitude* increase in difficulty. *Less* constrained problems appear to the *right* in the graphs: note that is the opposite of the convention for presenting results on random k-cnf (but is consistent with Achlioptas *et al.* (2000)). On the log scale we see that the most constrained instances are truly easy, because they are solved by forward-checking alone. The under-constrained problems are of moderate difficulty. This is due to the fact that the underconstrained problem are simply much larger after forward-checking. Again note that this situation is different from that of random k-cnf, where it is the over-constrained problems that are of moderate difficulty (Cook and Mitchell 1997).

At the peak the balanced QWH problems are much harder than the filtered QCP problems, showing that we have achieved the goal of creating a better benchmark for testing incomplete solvers. Balancing can be added to the filtered QCP model; the resulting balanced QCP model are yet more difficult. This indicates that balancing does not make QWH and QCP equivalent: the biases of the two approaches remain distinct. Both of the QWH models are harder than the QCP models in the under-constrained region to the right; we do not yet have an explanation for this phenomena.

Both the aligned and rectangle models are easy for Walksat, and show no hardness peak. In the over-constrained area (to the left) they require more flips to solve than the others. This is because clustering all the holes prevents arc consistency from completely solving the problems in the over-constrained region (there are more than one solutions), as it usually does for the other models.

Figure 6 shows the same ordering of hardness peaks for Satz. The behavior of Satz on the rectangle case is an interesting anomaly: it quickly becomes lost on the under-constrained problems and resorts to exhaustive search. This is because Satz gains its power from lookahead, and on under-constrained rectangular problems lookahead provides no pruning. Sato, which employs look-back rather than lookahead, does not exhibit this anomaly: it solves the rectangular problems as easily as the aligned ones.

We also measured the variance in the number of holes per row or column and the bandwidth of the balanced and random models. As expected, the variance was very low for the balanced case, averaging between 0.0 and 0.2 over the rage of ratios, compared with a range of 5.4 to 8.2 for the random case. Thus non-balanced problems will often have rows or columns that contain only a few, highly-constrained holes that can be easily filled in.

4 3-D and 2-D Encodings

Up to this point of the paper we have been concerned with understanding what makes a problem *intrinsically* hard. In practice, the difficulty of solving a particular instance using a particular algorithm is also dependent upon the details of the

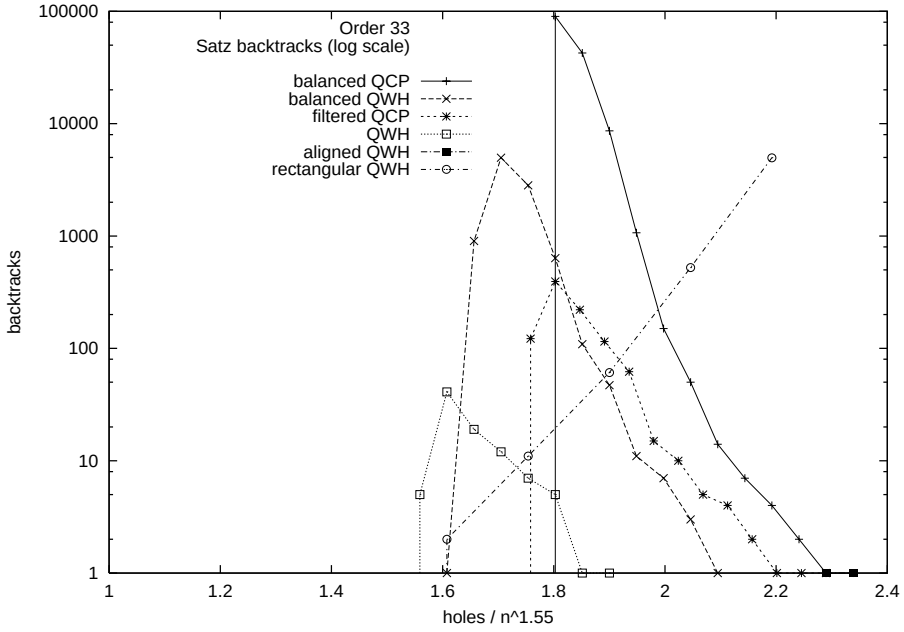

Figure 6: Comparison of generation models for order 33 using the Davis-Putnam type solver Satz. The x-axis is the percentage of holes (re-parameterized) and the y-axis is the number of backtracks (log scale). Each data point represents the median number of backtracks for 100 instances.

holes	holes/$n^{1.55}$	2-D	3-D
247	1.41	18	17
254	1.45	33	66
261	1.49	32	108
268	**1.53**	**23**	**109**
275	1.57	17	61
282	1.61	14	61
289	1.65	12	23

Table 2: Average number unit propagations performed immediately after each branch by Satz for the 2-D and 3-D encodings of order 28 QWH instances. Hardness peak is at $n^{1.55} = 1.53$ (261 holes).

representation of the problem.

Although quasigroup completion problems are most naturally represented as a CSP using multi-valued variables, encoding the problems using only Boolean variables in clausal form turns out to be surprisingly effective. Each Boolean variable represents a color assigned to a cell, so where n is the order there are n^3 variables. The most basic encoding, which we call the "2-dimensional" encoding, includes clauses that represent the following constraints:

1. Some color must be assigned to each cell;
2. No color is repeated in the same row;
3. No color is repeated in the same column.

Constraint (1) becomes a clause of length n for each cell, and (2) and (3) become sets of negative binary clauses. The total number of clauses is $O(n^4)$.

The binary representation of a Latin square can be viewed as a cube, where the dimensions are the row, column, and color. This view reveals an alternative way of stating the Latin square property: any set of variables determined by holding two of the dimensions fixed must contain exactly one true variable. The "3-dimensional" encoding captures this condition by also including the following constraints:

1. Each color much appear at least once in each row;
2. Each color much appear at least once in each column;
3. No two colors are assigned to the same cell.

As before, the total size of the 3-D encoding is $O(n^4)$.

As reported in Achlioptas *et al.* (2000), state of the art backtracking and local search SAT solvers using the 3-D encoding are competitive with specialized CSP algorithms. This is particularly surprising in light of the fact that the best CSP algorithms take explicit advantage of the structure of the problem, while the SAT algorithms are generic. Previous researchers have noted that the performance of backtracking CSP solvers on quasigroup problems is enhanced by using a dual representation (Slaney *et al.* 1995, Shaw *et al.* 1998, Zhang and Stickel 2000). This suggests a reason for the success of Davis-Putnam type SAT solvers: In the CSP dual encoding, there are variables for color/row pairs, where the domain is the set of columns, and similarly for color/column pairs, where the domain is the set of rows. The 3-D SAT encoding essentially gives us these dual variables and constraints for free.

This explanation is supported by the extremely poor performance of SAT solvers on the 2-D encodings of the problems. Neither Satz nor Sato can solve any instances at the hardness peak for orders larger than 28; using the 3-D encoding, by contrast, either could solve all instances with one backtrack on average. As shown in Figure 7, the work required by Satz explodes as the problem becomes underconstrained , requiring over 100,000 backtracks for order 28.

An explanation for the difference in performance of Satz on the different encodings can be found by examining the number of unit propagations triggered by each split in the search trees. Table 2 compares the number of unit propagations around the point at which the 2-D encodes become hard

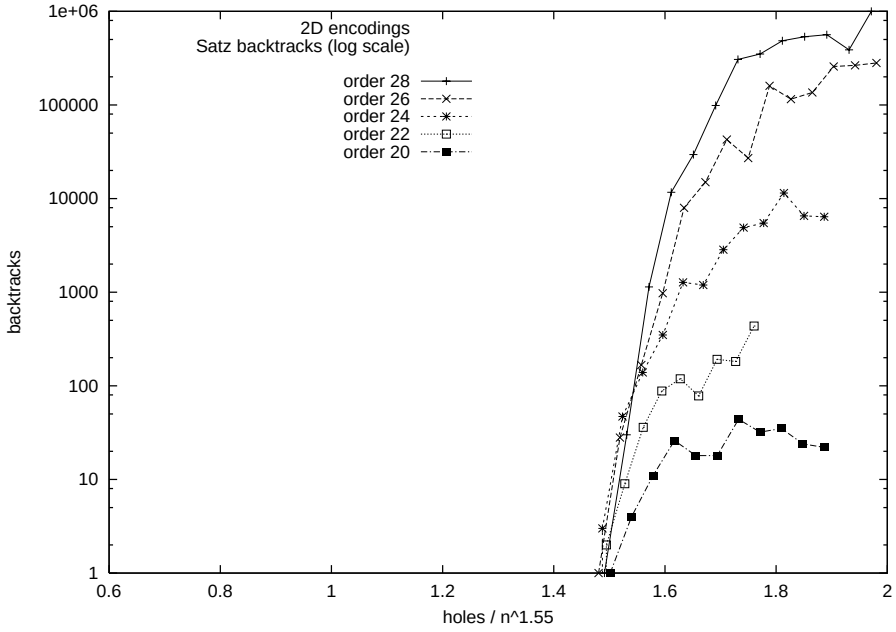

Figure 7: Comparison of 3-D versus 2-D encodings using Satz (backtracks, log scale) for 2-D encodings for orders 20 to 28. All problems of this size using the 3-D encodings could be solved by Satz in 0 or 1 backtracks.

for Satz (in bold). Note that at this point each split sets about 5 times more variables for the 3-D encoding than for the 2-D encodings.

Are the 2-D encodings inherently hard? Consider the performance of Walksat, on even larger orders (30 and 33), shown on in Figure 8. Walksat shows an unusual pattern: the 2-D encodings are somewhat *easier* than the 3-D encodings at the peak, and somewhat harder than then 3-D encodings in the under-constrained region to the right. Thus the 2-D and 3-D are in fact incomparable in terms of any general notion of hardness.

A significant difference between the 3-D and 2-D encodings is that for both Walksat and Satz it is difficult to see any hardness peak at the threshhold: the problems become hard and then stay at least as hard as they become more and more under-constrained. Note that the most underconstrained instances are *inherently* easy, since they correspond to a *empty* completion problem. This reinforces our argument that the 3-D encoding more accurately reflects the underlying computational properties of the quasigroup problem.

In summary, it is important to distinguish properties of a problem instance that make it inherently hard for all methods of attack and properties that make it accidently hard for particular methods. While the encoding style is such an accidental property, the main conjecture we present in this paper is that balance is an inherent property. The evidence for the conjecture is that increasing balance increases solution time for a *variety* of solvers.

5 Conclusions

Models of random problem generation serve two roles in AI: first, to provide tools for testing search and reasoning algorithms, and second, to further our understanding of what makes *particular* problems hard or easy to solve, as distinct from the fact that they fall in a class that is *worst-case* intractable. In this paper we introduced a range of new models of the quasigroup completion problem that serve these roles. We showed how a new notion of balance is an important factor in problem hardness. While previous work on balancing formulas considered the roles of positive and negative literals, our notion of balance is purely *structural*. Balancing improves the usefulness of the QWH model – one of the best models known for testing incomplete solvers – while retaining its formal simplicity and elegance.

References

D. Achlioptas, C. Gomes, H. Kautz, B. Selman (2000). Generating Satisfiable Instances. *Proc. AAAI-2000*.

Bayardo, R.J. and Schrag, R.C. (1996) Using csp look-back techniques to solve exceptionally hard sat instances. *Proc. CP-96)*, 46–60.

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

Cook, S.A. and Mitchell, D. (1997). Finding Hard Instances of the Satisfiability Problem: A Survey, in D. Du, J. Gu, and P. Pardalos, eds. *The Satisfiability Problem.* Vol. 35 of DIMACS Series in Discr. Math. and Theor. Comp. Sci., 1-17.

Fernandez de la Véga, W. (1981) On the bandwidth of random graphs. *Combinatorial Mathematics*, North-Holland, 633–638.

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

Gibbs, N.E. , Poole, Jr., W.E. and Stockmeyer, P.K. (1976). An algorithm for reducing the bandwidth and profile of a sparse matrix, *SIAM J. Numer. Anal.*, 13 (1976), 236–249.

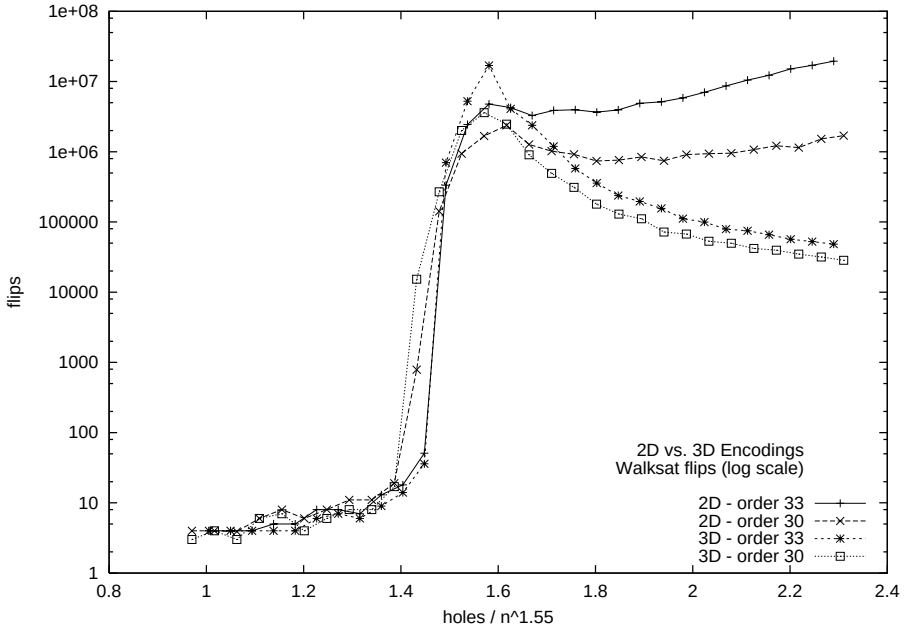

Figure 8: Comparison of 3-D versus 2-D encodings using Walksat (flips, log scale) for orders 30 and 33.

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

Hall, M. (1945). An existence theorem for latin squares. *Bull. Amer. Math Soc.*, 51, (1945), 387–388.

Hogg, T., Huberman, B.A., and Williams, C.P. (Eds.) (1996). Phase Transitions and Complexity. *Art. Intell.*, 81, 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 of one-way functions. *Proc. 21st STOC*.

Kirkpatrick, S. and Selman, B. (1994). Critical behavior in the satisfiability of 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.

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

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

Ryser, H. (1951). A combinatorial theorem with an application to latin rectangles. *Proc. Amer. Math. Soc.*, 2, (1951), 550-552.

Selman, B. and Levesque, H.J., and Mitchell, D.G. (1992). A New Method for Solving Hard Satisfiability Problems. *Proc. AAAI-92*.

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

Slaney, J., Fujita, M., and Stickel, M. (1995) Automated reasoning and exhaustive search: quasigroup existence problems. *Computers and Mathematics with Applications*, 29 (1995), 115-132.

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

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

Zhang, H. (1997). SATO: An Efficient Propositional Prover. Proc. CADE-97.

Zhang, H. and Stickel, M.E. (2000) Implementing the Davis-Putnam method. *Journal of Automated Reasoning* 24(1-2) 2000, 277-296.