Appears in *Proc. of the Second Int'l Conf. on Principles and Practice of Constraint Programming*, 46-60, 1996.

Using CSP Look-Back Techniques to Solve Exceptionally Hard SAT Instances

Roberto J. Bayardo Jr.
bayardo@cs.utexas.edu
http://www.cs.utexas.edu/users/bayardo/

Robert Schrag
schrag@cs.utexas.edu
http://www.cs.utexas.edu/users/schrag/

Dept. of Computer Sciences and Applied Research Laboratories
University of Texas at Austin
Taylor Hall, Rm. 141 (C0500)
Austin, TX 78712

Abstract. While CNF propositional satisfiability (SAT) is a sub-class of the more general constraint satisfaction problem (CSP), conventional wisdom has it that some well-known CSP look-back techniques -- including backjumping and learning -- are of little use for SAT. We enhance the Tableau SAT algorithm of Crawford and Auton with look-back techniques and evaluate its performance on problems specifically designed to challenge it.

The Random 3-SAT problem space has commonly been used to benchmark SAT algorithms because consistently difficult instances can be found near a region known as the phase transition. We modify Random 3-SAT in two ways which make instances even harder. First, we evaluate problems with structural regularities and find that CSP look-back techniques offer little advantage. Second, we evaluate problems in which a hard unsatisfiable instance of medium size is embedded in a larger instance, and we find the look-back enhancements to be indispensable. Without them, most instances are "exceptionally hard" -- orders of magnitude harder than typical Random 3-SAT instances with the same surface characteristics.

1 Introduction

Given the usual framework of backtrack search for systematic solution of the finite-domained constraint satisfaction problem (CSP), techniques intended to improve efficiency can be divided into two classes: *look-ahead* techniques, which exploit information about the remaining search space, and *look-back* techniques, which exploit information about search which has already taken place. The former class includes variable ordering heuristics, value ordering heuristics, and dynamic consistency enforcement schemes such as forward checking. The latter class includes schemes for backjumping (also known as intelligent backtracking) and learning (also known as nogood or constraint recording). In CSP algorithms, techniques from both classes are popular; for instance, one common combination of techniques (e.g. [1, 12, 28]) is forward checking, conflict-directed backjumping [23], and an ordering heuristic preferring variables with the smallest domains.

CNF propositional satisfiability (SAT) is a specific kind of CSP in which every variable ranges over the values $\{\text{true, false}\}$. For SAT, the most popular systematic algorithms are variants of the Davis-Logemann-Loveland modification [8] to the procedure originally defined by Davis and Putnam [7]; hereafter we refer to this procedure as “DP”. In CSP terms, the procedure is equivalent to backtrack search with forward checking and an ordering heuristic favoring unit-domained variables. Two effective modern variants of this algorithm are Tableau [5] and POSIT [11], both amounting to DP with highly optimized variable ordering heuristics. Are these SAT algorithms missing anything by not incorporating conflict-directed backjumping or another look-back technique? The standard Random 3-SAT problem space commonly used to benchmark SAT algorithms may not be a good place to look for the answer: Tableau is able to solve millions of instances from Random 3-SAT without any apparent trouble.

Here we challenge Tableau with modifications to Random 3-SAT to make instances more difficult. Random 3-SAT is already a source of consistently hard problem instances -- those in the region of the *phase transition* occurring when the ratio of constraints to variables increases through a critical value [27]. The phase transition separates an *under-constrained* region, where almost all instances are satisfiable and easy, from an *over-constrained* region, where almost all instances are unsatisfiable and relatively easy. We modify Random 3-SAT in two ways: first, we force problems to have structural regularities intended to confuse variable selection heuristics; second, we embed hard unsatisfiable instances into larger instances, making the unsatisfiability of the resulting instances difficult to identify. We also modify Tableau to incorporate some popular look-back techniques, and we evaluate the enhanced algorithm with the new problem spaces. In the case of highly regular problems, we find that look-back techniques offer little or no advantage; for solving our embedded problems, we find them indispensable.

Researchers working with random spaces for other CSPs [1, 17], with other SAT problem spaces [14, 15], or with Random 3-SAT but an algorithm other than Tableau [14, 15], have found rare instances in the under-constrained region so difficult as to render the mean difficulty higher there than in the transition region. Crawford and Auton [5] using Tableau and Random 3-SAT find no such “exceptionally hard” instances (EHIs). Compared to a reference problem space harder than Random 3-SAT (“Variable Regular 3-SAT” -- see Section 4), our embedding procedure generates instances that are “exceptionally hard” for Tableau -- orders of magnitude harder than other instances with the same surface characteristics. Our EHIs could as well result from Random 3-SAT, albeit with low probability. These instances have a clause to variable ratio that places them in the under-constrained region of the reference problem space. Given the difficulty and consistency with which they are generated, we believe they are useful as benchmarks for SAT algorithms.[1]

We find that look-back techniques greatly reduce the incidence of EHIs produced by our embedding procedure. The result is similar to that of Baker [1] who, using a random graph-coloring CSP space, finds no EHIs with respect to a conflict-directed backjumping and learning algorithm. In contrast, some instances produced by our embedding procedure remain difficult even for Tableau with conflict-directed back-

1 Implementations of the problem generators and algorithms defined in this paper are available through the Web page of the first author: http://www.cs.utexas.edu/users/bayardo/.

jumping and learning enhancements. Related work has identified other 3-SAT instances that are difficult for Tableau or other DP variants [16, 22]. The instances among these which we have tested are trivial for Tableau enhanced with CSP look-back techniques (see Section 6).

2 Definitions

SAT involves determining whether a given Boolean expression has a satisfying truth assignment. Any Boolean expression can be transformed to conjunctive-normal form (CNF) which allows a conjunction of clauses $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ where each *clause* C_i is a disjunction of *literals* $l_1 \vee l_2 \vee \ldots \vee l_i$. A literal is either a *variable* x_i or its *negation* $\neg x_i$, $1 \le i \le n$. Expressions in conjunctive normal form are easily seen to be instances of the CSP: each variable in the Boolean expression corresponds to a variable in the CSP with a Boolean domain, and each clause of i literals is a constraint disallowing exactly one truth assignment to the i variables mentioned. SAT restricted to conjunctive normal form with exactly k literals per clause is known as k-SAT. A common restriction of SAT that retains its NP-completeness is 3-SAT.

By *problem*, we mean an abstract description such as the definition for CSP, SAT, or 3-SAT which can be instantiated in different concrete ways -- e.g., by enumerating specific variables and constraints. By *instance*, we mean one of these particular instantiations. By *problem space*, we mean a parameterized set of problems, where each parameter represents a dimension of the space. Thus, one point in the larger space of 3-SAT is 3-SAT with exactly 75 variables and 325 clauses. In a *random problem space*, the probability distribution for the occurrence of a particular instance at any point depends on the operation of a non-deterministic procedure given the parameter values for that point as inputs. The procedure for the Random 3-SAT problem space is given below.

RANDOM 3-SAT: Inputs are the number of variables n and the number of clauses m. Three distinct variables are randomly selected out of the pool of n possible variables. Each variable is negated with probability $1/2$. These literals are combined to form a clause. m clauses are created in this manner and conjoined to form the 3-CNF Boolean expression.

For scaling across different problem sizes (different values of n), we use the *constraint ratio* m/n which is expressed in units of clauses per variable. Instances with high median difficulty can be found at the *crossover point*, occurring where half the generated instances are satisfiable. The crossover point may be thought of as the midpoint of the phase transition region. It turns out that the location of the crossover point is fairly stable in constraint ratio terms -- around 4.26 for Random 3-SAT [5].

3 Tableau and Look-Back Enhancements

We use Tableau [5] as our baseline SAT algorithm. Crawford and Auton show Tableau to be very effective at solving instances from Random 3-SAT and at overcoming the incidence of EHIs in the under-constrained region. For a full discussion of the heuristics which lead to its success, please see [5]. We create three look-back-enhanced versions of Tableau: one applying conflict-directed backjumping (CBJ) [23], another CBJ with third-order learning [12], and the last CBJ with unrestricted learning (sometimes referred to as "dependency-directed backtracking" [30]).

As look-back techniques, backjumping and learning are invoked when the algorithm reaches a failure point where at least one variable assignment must be undone before search can progress. Both exploit a set of "culprit" variables whose assignments are determined to be responsible for the failure. The method used to identify the set of culprits is critical in the effectiveness of the techniques. The culprit identification scheme used by Prosser's conflict directed backjumping is widely used [1, 12, 28], requires little overhead [28], and is provably more effective than some of its predecessors [20]. Given a culprit identification scheme, the next issue to be decided is how to exploit the culprits. Pure CBJ simply backs up to the most recent culprit to have been assigned a value without recording the culprit assignments. At the other extreme is unrestricted learning which records every assignment of culprit variables (called a *nogood*). A useful middle-ground is to apply CBJ and to record nogoods only if they are below a certain size. For instance, third-order learning [12] records only those nogoods mentioning three or fewer variables. In the SAT context, this corresponds to recording derived clauses of three or fewer literals.

In the experiments that follow, we concentrate on CBJ and bounded learning enhancements of Tableau. We experiment briefly with unrestricted learning, but find it too expensive on the more difficult instances.

4 Regularity-Inducing 3-SAT Generators

Tableau and other modern SAT algorithms exploit irregularities within the search space to realize inevitable dead-ends as quickly as possible. We were interested in identifying the effects of highly regular instances on Tableau and its enhancements. Various other studies [6, 13, 29, 31] have investigated the effects of increased regularity on SAT and CSP solving, finding that higher regularity increases mean difficulty. While look-back techniques do not significantly improve Tableau's mean performance on the instances below, we discover interesting phase transition properties which we exploit to develop a harder problem generator in the following section.

We first define two new generation procedures based on Random 3-SAT that progressively eliminate certain sources of irregularity. An obvious potential irregularity within a Random 3-SAT instance is that some variables may appear more often than others. The following generation procedure removes this irregularity nearly completely:

VARIABLE-REGULAR 3-SAT: Inputs are the number of variables (n) and the number of clauses (m). The instance is constructed by putting $\lfloor 3(m/n) \rfloor$ occurrences of each variable in a "bag". A random set of unique variables is then added to the bag so that there are exactly $3m$ variables in it. To construct each clause, three distinct variables are removed from the bag. Each variable is negated with probability $1/2$ to form a clause, and clauses are conjoined to form the 3-CNF expression. If there are only one or two distinct variables remaining within the bag, additional distinct variables are selected randomly from the set of all variables.

Since variables are negated at random in Variable-Regular 3-SAT, a given variable may appear negated more often than not (or vice versa). The next generation procedure removes this source of irregularity nearly completely. This problem space is equivalent to the "doubly balanced" SAT space investigated independently by Dubois and Boufkhad [10], and similar to those defined by Genisson and Sais [13].

LITERAL-REGULAR 3-SAT: Inputs are the number of variables (n) and the number of clauses (m). There are $2n$ possible literals given n variables, so $\lfloor 3m/2n \rfloor$ occurrences of each literal are placed in a bag. A random set of unique literals is then added to the bag so that there are exactly $3m$ literals in it. To construct each clause, three literals on distinct variables are removed from the bag. If there are only 1 or 2 distinct variables mentioned in literals remaining the bag, additional distinct variables are randomly selected from the set of all variables and negated with probability $1/2$.

Data on the location of the phase transition and mean problem difficulty with respect to Tableau for instances generated by the regularity-inducing generators appear in Figure 1. Each point plotted in both graphs results from 500 instances solved by our implementation of Tableau.

While both generators increase regularity, one exhibits a phase transition to the right of Random 3-SAT's, and the other to the left. The first graph in the figure displays the phase transition properties for each procedure when n is fixed at 140. Smith and Dyer [29] find a similar rightward shift with CSPs when decreasing constraint-graph regularity. They point out that it is more difficult to assign a value to a highly constrained variable than to a less constrained one; thus, greater variability in constraint graph degree should lead on average to greater frequency of unsatisfiability for given n and m. This helps to explain why variable-regularity shifts the phase transition to the right from Random 3-SAT. Genisson and Sais [13] reported a similar leftward shift with their literal-regular 3-SAT generator. We believe that literal-regular instances typically require fewer clauses for unsatisfiability because the balance of positive and negative variable occurrences provides more opportunities for resolution, leading to a greater probability of ultimately deriving the empty clause.

Also displayed in Figure 1 are mean problem difficulty at various values of input parameters. Difficulty is represented by the number of branch points encountered by Tableau. *Branch points* are defined as search tree nodes where Tableau does a significant amount of work (i.e. more than just unit propagation) in branching on both possible truth values [5]. As expected, mean difficulty at crossover increased with regularity. Literal-Regular 3-SAT exhibits the highest mean difficulty; at this relatively low value of n, its peak is almost an order of magnitude higher than that of Random 3-SAT.

We repeated the experiments at larger and smaller values of n in order to see how the phase transition and mean difficulty changed, this time averaging over 1000

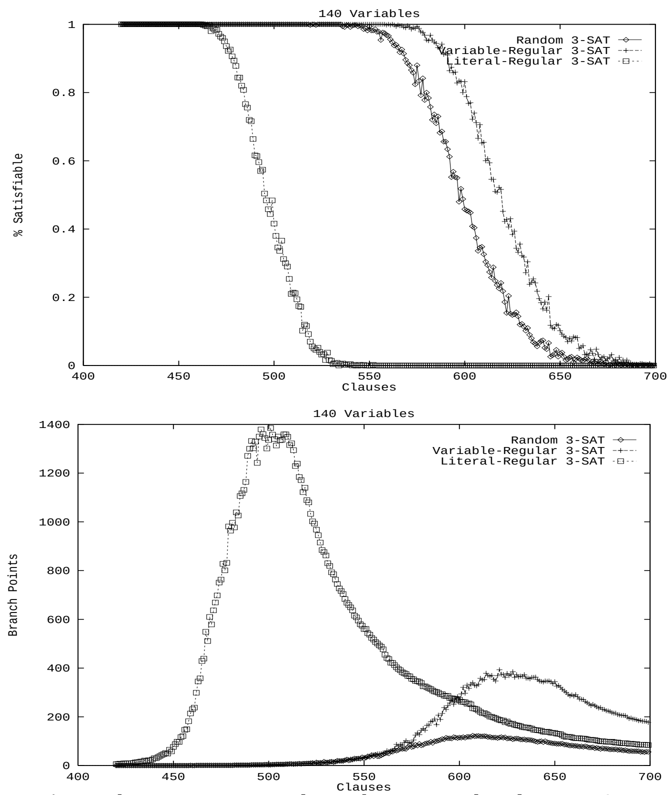

Fig. 1. Phase transitions and search space explored at n=140.

instances per data point. Recall that for Random 3-SAT the location of the crossover point is fairly stable at a constraint ratio near 4.26. The graphs in Figure 2 illustrate the crossover points for the other problem spaces, and the respective difficulty at the crossover point. Crossover point locations for these new generators are also stable in constraint ratio terms. We derived a crossover point constraint ratio of 4.41 for Variable-Regular 3-SAT and 3.54 for Literal-Regular 3-SAT.

Measuring difficulty in branch points explored by Tableau, it is known that difficulty of crossover point problems from Random 3-SAT approximately doubles every time the number of variables is increased by 20 (at least up to $n = 300$) [5]. For Variable-Regular 3-SAT, we see that difficulty increases by a factor of 2.4 with an increment of 20 variables (within the range explored). For Literal-Regular 3-SAT, the difficulty increases with a factor of approximately 2.9.

We added conflict-directed backjumping and third-order learning to Tableau, anticipating that learning (which derives new clauses during search) could create irregularities for the variable ordering heuristics to exploit. We found that neither scheme improved runtime nor reduced search space explored beyond a few percentage points. Tableau, while performing worse on these instances than typical Random 3-SAT instances, has a respectable growth rate when compared to naive DP variants. For instance, Crawford and Auton [4] find that DP using a most-constrained-first variable

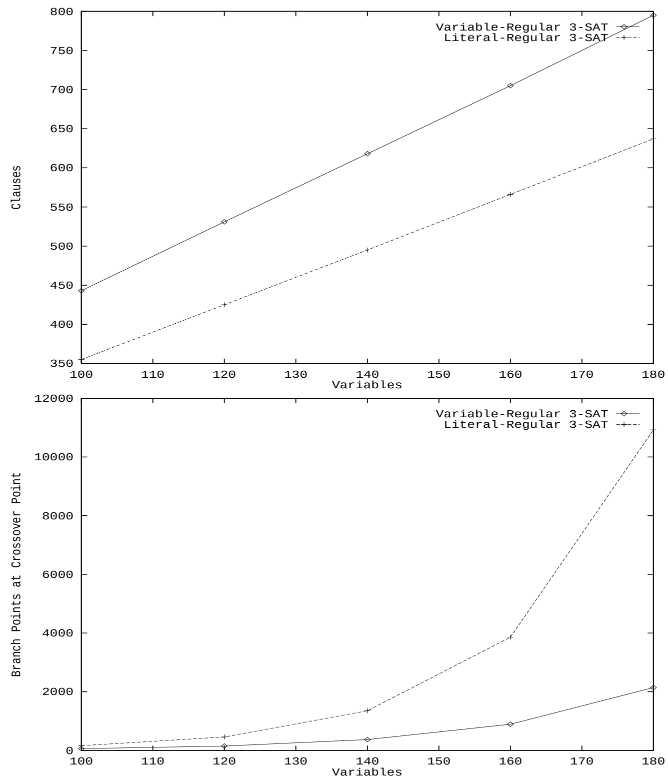

Fig. 2. Crossover point locations and difficulty at the crossover point.

selection heuristic requires over 10,000 branch points for Random 3-SAT problems with as few as 100 variables. After assigning a few variables, we suspect enough irregularities appear in the resulting sub-problems to make Tableau's look-ahead heuristics effective. The irregularities created by learning were not significant in comparison.

5 Manufacturing and Solving Exceptionally Hard Problems

Several researchers [1, 14, 15, 17] have found rare instances in the under-constrained region of various problem spaces so difficult as to render the mean difficulty higher than that of instances from the crossover point. Some of these instances have been found to contain small unsatisfiable sub-problems [15]. In this section, instead of randomly generating instances from the under-constrained region of a particular problem space in search of exceptionally hard instances, we actively generate them by creating under-constrained instances with small unsatisfiable sub-problems.

A simple approach for generating an under-constrained instance containing an unsatisfiable sub-problem is to take the union of the clauses from an under-constrained instance and an unsatisfiable one. However, most instances created using this approach have surface characteristics that render them easy. For example, if the two combined instances consist of disjoint variables, a simple connected components algorithm could be used to identify each sub-problem for their independent solution. Without such a preprocessing phase, we have found that Tableau can perform poorly on such instances if the variable occurring most frequently appears in the under-constrained instance: Tableau will initially attempt to solve the under-constrained instance, and for each solution, it attempts (and fails) to solve the unsatisfiable one. The addition of conflict-directed backjumping completely remedies this behavior, since it allows each sub-problem to be effectively solved independently in conjunction with the most-constrained first variable ordering heuristic. If the variable names of the combined instances are overlapping, then variables shared between the instances almost always occur most frequently (unless additional steps are taken to prevent this). This causes Tableau to branch initially on those variables, allowing it to determine unsatisfiability without exceptional difficulty.

In this section, we show how the phase transition characteristics of the regular problem spaces can be exploited to conveniently generate under-constrained instances containing well-concealed small unsatisfiable sub-problems. The procedure is used to generate instances whose constraint ratios suggest they should be easily satisfiable with reference to another problem space. Instead, they frequently turn out to be exceptionally hard. We find that look-back enhancements provide a substantial degree of insurance against poor performance, though they fail to eliminate it completely.

In principle, an instance can be "exceptionally hard" only with reference to a given algorithm and a particular problem space. For example, looking at Figure 2b, crossover instances from Literal-Regular 3-SAT would be considered hard with reference to Random 3-SAT. As a convention, we take an EHI to be any instance which requires at least an order of magnitude more work than the mean difficulty of crossover instances in the reference problem space. Here, we use Variable-Regular 3-SAT as our reference space since the generator produces only variable regular instances. In our experiments with Variable-Regular 3-SAT and those with Random 3-SAT, we found no instances at crossover requiring more than 5 times the mean number of branch points, so our EHIs are much harder than any of the observed crossover instances.

The procedure below conceals a problem instance within a larger one. It uses the Variable-Regular 3-SAT procedure to embed the input instance P' with n' variables and m' clauses within a larger randomly generated instance of size n, m.

EMBEDDING PROCEDURE: Inputs are an instance P' of size n', m', and parameters n, m specifying the size of the instance to be generated. The number of occurrences of any variable in P' is required to be no more than $\lfloor 3(m/n) \rfloor$. The variables of P' are first renamed randomly to variables within the set of n variables to appear in the generated instance. Next, a bag is filled with variable occurrences exactly as is done by Variable-Regular 3-SAT with parameters n, m. Then, for every occurrence of a variable appearing in the renamed m' clauses, an occurrence of that variable is removed from the bag. Afterwards, the remaining $m - m'$ clauses are generated as defined by the Variable-Regular instance generator.

If P' is unsatisfiable, then the resulting problem will also be unsatisfiable since it contains the (renamed) clauses of P'. The procedure below uses the technique to create unsatisfiable variable-regular instances.

EXCEPTIONALLY HARD 3-SAT: Inputs are four parameters, n, m, n', m'. We begin by using the Literal-Regular 3-SAT procedure to generate an instance P' of size n', m'. The procedure is invoked until an unsatisfiable instance is produced. Then, clauses are greedily removed at random from P' until there is no single clause that can be removed without rendering the instance satisfiable. At this point, the reduced P' is checked to ensure no variable appears more than $\lfloor 3(m/n) \rfloor$ times. If a variable appears too often, then we start over. Otherwise, we embed P' into a size n, m instance using the previously-described embedding procedure.

The above procedure must find an unsatisfiable instance that can be effectively concealed by way of low variable occurrences. Its strategy for maximizing the probability of concealment is as follows. First, it selects an unsatisfiable instance from Literal-Regular 3-SAT. On average, these require fewer clauses for unsatisfiability than instances from Random 3-SAT and Variable-Regular 3-SAT due to the left-shifted phase transition. The procedure then applies the greedy reduction phase to make the instance even easier to hide (we find that reduction typically removes 20-40% of the clauses from unsatisfiable crossover instances).[1] In the embedding phase, the reduced instance is padded with clauses that suffice to make the result a possible output of Variable-Regular 3-SAT. The right-shifted phase transition of Variable-Regular 3-SAT allows padding with more clauses than Literal-Regular or Random 3-SAT for producing what superficially appear to be under-constrained instances.

The following tables report the performance of Tableau and its enhancements on instances from Exceptionally Hard 3-SAT. Tableau is denoted by “Tab”, Tableau with conflict-directed backjumping “Tab+CBJ” and Tableau with conflict-directed backjumping and third-order learning “Tab+CBJ+lrn”. We continue to report problem difficulty in terms of branch points because overhead of these additional enhancements was small. The overhead of conflict-directed backjumping alone was negligible (less than 3%) since Tableau expends most of its effort selecting branch variables. Learning derives new clauses which had to be tested even by the branch-variable selection procedure. Third-order learning, however, typically recorded only a few clauses, keeping overhead well within 20% even on the hardest problems. At the end of this section, we discuss preliminary experiments with unrestricted learning.

We used a constraint ratio of 3.5 or less for determining m for the various values of n since it is well within the under-constrained region of Variable-Regular 3-

1 This phase is NP-hard, though it presents no practical problem as long as we choose to embed small instances. For greater efficiency, we could embed an instance selected from a set of pre-reduced instances instead of generating and reducing a new instance for embedding with each invocation of the Exceptionally Hard 3-SAT procedure.

SAT. For n', m', we used the formula $n = 3.5m + 5$ to produce hard Literal-Regular 3-SAT instances for embedding.

To facilitate experiments with large numbers of instances (10,000 per value of n), we had the algorithm halt and report failure beyond a threshold of branch points an order of magnitude or more than the mean required for Variable-Regular crossover instances with the same number of variables (10,000 branch points for $n = 75$ and $n = 140$, 50,000 branch points for $n = 200$). We also report the mean difficulty of a smaller number (200) of problems on which the failure mechanism was turned off.

Table 1. Exceptionally hard problem statistics for n=75, m=225, n'=10, m'=40

Algorithm	**Mean Difficulty of Solved Instances (branch points)**	**Failure Rate (%) [> 10000]**	**Mean Difficulty of 200 instances (branch points)**
Tableau	2,672	66.5	87,993
Tab + CBJ	113	0	113
Tab + CBJ + lrn	3	0	3

Table 1 shows that unenhanced Tableau performs poorly even on very small instances from Exceptionally Hard 3-SAT. For Variable -Regular crossover instances with $n = 75$, Tableau requires a mere 20 branch points on average. Performance on 75 variable Exceptionally Hard 3-SAT instances is over 3 orders of magnitude worse. While Tableau with backjumping is effective at solving these problems, the addition of learning makes them trivial. For problems barely beyond $n = 75$, we found that Tableau's failure rate rapidly approached 100%.

Table 2. Exceptionally hard problem statistics for n=140, m=490, n'=20, m'=75

Algorithm	**Mean Difficulty of Solved Instances (branch points)**	**Failure Rate (%) [> 10000]**	**Mean Difficulty of 200 instances (branch points)**
Tab + CBJ	1,904	57.2	55,122
Tab + CBJ + lrn	12	0	12

The next data point (Table 2) illustrates that Tableau with conflict-directed backjumping alone begins to go awry at large enough problems. Others [15, 28] have also found that backjumping alone fails to eliminate occurrence of exceptionally hard instances, though in the context of sparse CSP or much larger SAT instances.

Table 3. Exceptionally hard problem statistics for n=200, m=700, n'=30, m'=110

Algorithm	Mean Difficulty of Solved Instances (branch points)	Failure Rate (%) [> 50000]
Tab + CBJ + lrn	78	0

Table 3 shows that the learning algorithm remains completely effective even at larger problem sizes when the embedded instance is small. The failure cutoff is increased to 50,000 branch points for these larger problems since the average Variable-Regular crossover instance of 200 variables requires approximately 5000 branch points. It appears, however, that if the embedded and actual instances are large enough, we can elicit failure even in the learning algorithm (Table 4 below). The failure rate remains relatively low at the data points we investigated. Further experimentation is required in order to determine if the failure rate approaches 100% with larger problems (as is clearly the case with the other two algorithms).

Table 4. Exceptionally hard problem statistics for n=200, m=700, n'=50, m'=180

Algorithm	Mean Difficulty of Solved Instances (branch points)	Failure Rate (%) [> 50000]
Tab + CBJ + lrn	3,702	3.4945

We have performed some experiments with unrestricted learning, but have been unable to draw any solid conclusions about its effects other than that its overhead becomes unacceptably high on sufficiently large and dense SAT instances. For example, on difficult instances from Exceptionally Hard 3-SAT, the third-order learning algorithm was 30 times faster in terms of branch points searched per second. On 50 instances from the <200, 700, 50, 180> point, the hardest instance found for the unrestricted learning algorithm required 22,405 branch points. This translates to well over 10 times the CPU time that the third-order learning algorithm required to reach failure. To account for this overhead, we feel the definition of an exceptionally hard instance for unrestricted learning algorithms should take CPU time into account. By such a definition, our implementation of unrestricted learning fails to eliminate them completely.

Baker [1] and Frost and Dechter [12] found that the overhead of their unrestricted learning algorithms was not excessive. We believe the our different findings are primarily due to the constraint density of SAT compared to binary CSP. 3-SAT instances require many constraints (clauses), since each excludes only a small fraction of potential truth assignments. Further, each constraint is defined on three variables instead of two. As a result, the set of variables responsible for each failure when solving a SAT instance is often large. Another potential cause for our different findings is that some instances produced by Exceptionally Hard 3-SAT required extensive search even of the unrestricted learning algorithm. Baker's instances were easy for his unrestricted learning algorithm, so it never had the opportunity to derive an excessive number of constraints. We believe that any SAT algorithm applying learning on instances like ours will require either limited order learning as employed here, time or relevance limits on derived clauses [2,16], or some method for efficiently producing smaller culprit sets than conflict-directed backjumping (e.g. possibly along the lines of Dechter's deep learning schemes [9]).

6 Related and Future Work

Ginsberg & McAllester [16] evaluate a CSP algorithm they call "partial-order dynamic backtracking" on a 3-SAT problem space with restricted structure. This problem space creates an instance by arranging the variables on a two-dimensional grid and creating clauses that contain variables forming a triangle with two sides of unit Euclidean length. The algorithm incorporates look-ahead techniques not specifically geared for SAT and look-back techniques similar to CBJ and learning. They find it immune to pathologies encountered by Tableau on these instances. The hardest problems used in their evaluation were crossover instances from their new SAT problem space with 625 variables. We found these instances trivial for Tableau with CBJ and third-order learning, requiring on average 6 and at maximum 28 branch points on the 10,000 instances we attempted. We found that CBJ-enhanced Tableau has occasional difficulties on these same instances, requiring less than 22 branches on over half the instances, but over 50,000 branches on 2.51% of them. This suggests these instances may have properties similar to (though not as pronounced as) those from Exceptionally-Hard 3-SAT.

Mazure et al. [22] recently developed a look-ahead technique for DP based on GSAT [26]. The technique selects branch variables by counting the number of times a variable occurs in clauses falsified by assignments made during a GSAT search on the current sub-problem. The intent is to focus search on variables that may be part of an inconsistent kernel. They evaluate their technique on several DIMACS benchmark instances[1] which were infeasible for DP. Some of the "`AIM`" instances from this suite that are difficult for DP are trivial for their algorithm. We found that all of the largest (200 variable) "`AIM`" instances were trivial for Tableau with CBJ and learning, the most difficult requiring 27 branch points. Some of these instances were difficult for Tableau without learning but with CBJ. We have not yet explored the effect of their technique on our problem spaces.

Lee and Plaisted [21] enhance DP with a backjumping scheme similar to (but not as powerful as) CBJ which they use as a subroutine in a first-order theorem proving system. Gent and Walsh [15] experiment with an implementation of this algorithm[2] on a SAT problem space they call "Constant Probability" and find that the backjumping scheme reduces the incidence of EHIs in the under-constrained region, but fails to eliminate them at large enough problem sizes. Chvatal [3] also applies look-back to SAT through "resolution search" -- a DP variant with what appears to be a novel learning scheme.

---

1 These instances are available through anonymous FTP at ftp:dimacs.rutgers.edu within directory pub/challenge/sat/benchmarks/cnf.

2 This is the "intelligent backtracking" algorithm whose implementation they credit to Mark Stickel.

Others have developed procedures intended to generate hard random problems. Iwama et al. [19] and Rauzy [24] independently generate 3-SAT instances which are known in advance to be satisfiable or unsatisfiable. The authors do not directly compare the difficulty of these problems to problems from Random 3-SAT. Genisson and Sais [13] investigate 3-SAT generators with controlled distributions of literals -- much like our Literal-Regular 3-SAT -- and also find a left-shifted phase transition and increased difficulty. Dubois and Boufkhad also investigate literal-regular instances they call “doubly balanced” in a forthcoming paper [10]. Culberson et al. [6] and Vlasie [31] describe generators for graph coloring CSPs in which graphs are endowed with a variety of structural properties intended to make coloring difficult. Their empirical results cannot be compared to ours directly, but they also find that increased regularity increases mean problem difficulty. Whether look-back techniques are important for these problem spaces remains to be evaluated empirically.

Crawford and Auton [5] were unable to find any EHIs for Tableau in the underconstrained region of Random 3-SAT. Our results with Exceptionally Hard 3-SAT show that they do exist, albeit in Random 3-SAT with low probability. Gent and Walsh report that an improved DVO heuristic [14] and improved constraint propagation method [15] (both look-ahead techniques) fail to eliminate EHIs for their 3-SAT generators and DP implementation. Baker [1], working with graph coloring CSPs, finds no problems to be extremely hard for a conflict-directed backjumping and unrestricted learning algorithm. Selman and Kirkpatrick [25], using an earlier version of Tableau [4] and Random 3-SAT, investigate the incidence of EHIs when a given instance is subject to an equivalence-preserving random renaming of its variables. They report observing the same incidence of EHIs whether running Tableau on 5000 different permutations of the same 20 source instances, or whether sampling 5000 instances independently. This suggests these EHIs arise on account of unfortunate variable orderings. We have not yet investigated the effects of random renamings on our instances. The fact that they occur with near certainty for unenhanced Tableau when generating sufficiently large problems suggests that the source of their difficulty may be qualitatively different.

The fact that our generated EHIs are unsatisfiable means that sound but incomplete algorithms such as GSAT [26] cannot be used to address them. We are considering a related problem generator of satisfiable instances for the purpose of benchmarking sound-and-incomplete SAT algorithms. We embed hard satisfiable instances instead of unsatisfiable ones by removing one additional clause immediately following the reduction phase of Exceptionally Hard 3-SAT. Limited experimentation has shown similar (though not as pronounced) difficulties result for unenhanced Tableau, even though the resulting instances are almost always satisfiable.

7 Conclusions

We have shown that, contrary to the conventional wisdom, CSP look-back techniques are useful for SAT. We were able to significantly reduce the incidence of exceptionally hard instances (EHIs) encountered by the Tableau SAT algorithm after enhancing it with look-back techniques. We devised a new generator which usually succeeds in producing instances which are exceptionally hard for unenhanced Tableau when compared to the difficulty of other common benchmark instances. Relatively few of these instances remained exceptionally hard for the look-back-enhanced versions of Tableau.

It may be that look-back techniques are essential to solving these instances efficiently, though we encourage experimentation with non-look-back algorithms which might refute this hypothesis. At the same time, we do not wish to minimize the significance of look-ahead techniques. Tableau's variable selection heuristics make it competitive with the best current SAT algorithms, and we would expect to encounter many more exceptionally hard instances without them. We do not believe that any generally effective SAT algorithm can totally eradicate the incidence of EHIs. We do believe that selected look-ahead and look-back techniques with modest overhead can provide some valuable insurance against them.

Acknowledgments

We wish to thank Jimi Crawford for the Tableau executable and assistance in replicating its behavior. We also thank Richard Genisson, Ian Gent, Daniel Miranker, David Mitchell, Lakhdar Sais, Mark Stickel and Toby Walsh for their comments and assistance.

References

1. A. B. Baker, *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results*. Ph.D. Dissertation, Dept. of Computer and Information Science, University of Oregon, March 1995.
2. R. J. Bayardo, A complexity analysis of space-bounded learning algorithms for the constraint satisfaction problem, In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, 1996.
3. V. Chvatal, Resolution search. *Discrete Applied Math*, to appear, 1996.
4. J. M. Crawford and L. D. Auton, Experimental results on the crossover point in satisfiability problems. *AAAI-93*, 21-27, 1993.
5. J. M. Crawford and L. D. Auton, Experimental results on the crossover point in random 3SAT. *Artificial Intelligence* 81(1-2), 31-57, 1996.
6. J. Culberson, A. Beacham and D. Papp, Hiding our colors, Workshop on Studying and Solving Really Hard Problems, International Conference on Principles and Practice of Constraint Programming, 1995.
7. M. Davis and H. Putnam, A computing procedure for quantification theory, *JACM* 7, 201-215, 1960.
8. M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, *CACM* 5, 394-397, 1962.
9. R. Dechter, Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence* 41(3):273-312, 1990.
10. O. Dubois and Y. Boufkhad, From very hard doubly balanced SAT formulae to easy unbalanced SAT formulae, variations of the satisfiability threshold, *Proceedings of the DIMACS workshop on the Satisfiability Problem: Theory and Applications*, Ding-Zhu Du, Jun Gu, and Panos Pardalos, eds., March 1996.

11. J. W. Freeman, *Improvements to Propositional Satisfiability Search Algorithms*. Ph.D. Dissertation, Dept. of Computer and Information Science, University of Pennsylvania, 1995.
12. D. Frost and R. Dechter, Dead-end driven learning. In *Proceedings of AAAI-94*, 294-300, 1994.
13. R. Genisson and L. Sais, Some ideas on random generation of k-SAT instances, AAAI-94 Workshop on Experimental Evaluation of Reasoning and Search Methods, 1994.
14. I. Gent and T. Walsh, Easy problems are sometimes hard, *Artificial Intelligence* 70, 335-345, 1994.
15. I. Gent and T. Walsh, The satisfiability constraint gap. *Artificial Intelligence* 81(1-2), 59-80, 1996.
16. M. Ginsberg and D. McAllester, GSAT and dynamic backtracking, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference*, 226-237, 1994.
17. T. Hogg and C. P. Williams, The hardest constraint problems: A double phase transition. *Artificial Intelligence* 69, 359-377, 1994.
18. T. Hogg, B. A. Huberman, C. Williams (eds.), *Artificial Intelligence* 81(1-2), Special issue on Frontiers in Problem Solving: Phase Transitions and Complexity, Elsevier, March 1996.
19. K. Iwama, H. Abeta, E. Miyano, Random generation of satisfiable and unsatisfiable CNF predicates, in *Algorithms, Software, Architecture, Information Processing 92* volume 1, 322-328, 1992.
20. G. Kondrak and P. van Beek, A theoretical evaluation of selected backtracking algorithms, *IJCAI-95*, 541-547, 1995.
21. S.-J. Lee and D. A. Plaisted, Eliminating duplication with the hyper-linking strategy, *Journal of Automated Reasoning* 9, 25-42, 1992.
22. B. Mazure, L. Sais and E. Gregoire, Detecting logical inconsistencies, In *Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics*, 116-121, 1996.
23. P. Prosser, Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3), 268-299, 1993.
24. A. Rauzy, *On the Random Generation of 3-SAT Instances*, Technical Report 1060-95, Laboratoire Bordelais de Recherche en Informatique, Universite Bordeaux I, 1995.
25. B. Selman and S. Kirkpatrick, Critical behavior in the satisfiability of random Boolean expressions II: Computational cost of systematic search, *Artificial Intelligence* 81(1-2), 273-295, 1996.
26. B. Selman, H. Levesque, and D. Mitchell, A new method for solving hard satisfiability problems, *AAAI-92*, 440-446, 1992.
27. B. Selman, D. Mitchell, and H. Levesque, Generating hard satisfiability problems, *Artificial Intelligence* 81(1-2), 17-29, 1996.
28. B. A. Smith and S. A. Grant, Sparse constraint graphs and exceptionally hard problems, *IJCAI-95*, 646-651, 1995.
29. B. M. Smith and M. E. Dyer, Locating the phase transition in binary constraint satisfaction problems, *Artificial Intelligence* 81(1-2), 155-181, 1996.
30. R. M. Stallman and G. J. Sussman, Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence* 9, 135-196, 1977.
31. R. D. Vlasie, Systematic generation of very hard cases for graph 3-colorability, In *Proceedings of ICTAI-95*, IEEE Press, 1995.