Unsatisfied Variables in Local Search

Ian P. Gent*
Department of AI
University of Edinburgh
80 South Bridge, Edinburgh
United Kingdom
ipg@cs.strath.ac.uk

Toby Walsh
Mechanized Reasoning Group
IRST, Trento &
DIST, University of Genoa,
Italy
toby@irst.it

Abstract

Several local search algorithms for propositional satisfiability have been proposed which can solve hard random problems beyond the range of conventional backtracking procedures. In this paper, we explore the impact of focusing search in these procedures on the "unsatisfied variables"; that is, those variables which appear in clauses which are not yet satisfied. For random problems, we show that such a focus reduces the sensitivity to input parameters. We also observe a simple scaling law in performance. For non-random problems, we show that whilst this focus can improve performance, many problems remain difficult. We speculate that such problems will remain hard for local search unless constraint propagation techniques can be combined with hill-climbing.

1 Introduction

Local search is often surprisingly effective as a semi-decision procedure for many NP-hard problems. For example, GSAT, a greedy random hill-climbing procedure for propositional satisfiability (or SAT) is very good at solving hard random problems [16]. Given a formula in conjunctive normal form,[1] GSAT computes a randomly generated truth assignment, and hill-climbs by repeatedly flipping the variable assignment which most increases the number of clauses satisfied. If there is a choice between several equally good flips, GSAT picks one at random. If there are no upward flips, GSAT makes a sideways flip. Without sideways flips, the performance of GSAT degrades greatly. In [5] it is shown that much of the search consists of the exploration of large "plateaus" where sideways flips predominate and only the occasional up flip is possible.

Occasional downward flips appear to improve performance. A variant of GSAT, called GSAT with random walk [14] makes downward flips even when up flips are possible. With probability p, GSAT with random walk flips a variable in an unsatisfied clause, and otherwise hill-climbs normally. Flipping a variable in an unsatisfied clause can actually decrease the number of satisfied clauses. Nevertheless random walk improves the performance of GSAT considerably. While the fastest complete procedures

*Current address: Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, United Kingdom

[1] A formula in conjunctive normal form is a conjunction of clauses, where each clause a disjunction of literals and each literal is a negated or un-negated variable.

for SAT can solve hard random 3-SAT problems (these are defined in §3) up to about 400 variables, GSAT with random walk has solved instances with 2000 variables [15].

GSAT with random walk focuses attention on variables in unsatisfied clauses (called, from now on, unsatisfied variables). This appears to be a better strategy than merely flipping a variable *at random* with probability p [15]. In all models, at least one of the variables in an unsatisfied clause must have the opposite truth value. We must therefore flip at least one of them en route to a model. Flipping an unsatisfied variable is also guaranteed to change the set of unsatisfied clauses. Since GSAT's search is governed by the variables in this set, flipping an unsatisfied variable introduces diversity into the search. For these two reasons, focussing on the unsatisfied variables may move us to more "interesting" parts of the search space from where we can hill-climb to a model.

In this paper, we report extensive experimental investigations into the importance of flipping unsatisfied variables. We identify the ways in which the addition of random walk improves GSAT. Surprisingly, the major contribution appears not to be better run times but a reduction in sensitivity to input parameters. We also add random walk to some of the variants of GSAT introduced in [4], some of which outperform GSAT with walk. Finally, we introduce a new procedure called JumpSAT in which flipping unsatisfied variables is made paramount. Given recent concern about the representativeness of random problem classes, we use both random and non-random problems.

2 A Generalized Hill-climbing Procedure

We present our work within the framework of a generalized hill-climbing procedure called GENSAT introduced in [4] and given in Figure 1. This procedure embodies the common features found in a wide range of hill-climbing procedures for satisfiability like GSAT. Although the name GSAT has been used for both GSAT and GSAT with random walk, here we distinguish between the two variants by calling GSAT with random walk by the name GRSAT. Both GSAT and GRSAT are instances of GENSAT.

As in [1], the select function has been generalized from that used in [4] to allow for the possibility of a random walk option. GENSAT now has four parameters: Σ, Max-tries, Max-flips and p. Σ is the formula to satisfy; Max-tries is the number of restarts and is usually set as large as patience allows; Max-flips is the maximum number of flips before a restart; and p is the probability of performing a random walk. GSAT is a particular instance of GENSAT in which there is no random walk (*i.e.* $p = 0$), *initial* generates a random truth assignment, *hclimb* returns those variables whose truth assignment if flipped gives the greatest increase in the number of clauses satisfied (called the "score" from now on) and *pick* chooses one of these variables at random. A detailed experimental investigation of GENSAT variants without random walk is reported in [4].

3 Random Problems

Our initial experiments use the random k-SAT problem class. This class was used in earlier experiments with GSAT [16, 4] and in many studies of complete procedures for satisfiability like Davis-Putnam [2, 11, 7, 8]. A problem in random k-SAT consists of L clauses, each of which has k literals chosen uniformly from the N variables, each literal being positive or negative with probability $\frac{1}{2}$.

For random 3-SAT, there is a phase transition between satisfiability and unsatisfiability for L/N≈4.3. Problems in the phase transition are typically much more difficult to solve than problems away from the transition [2]. The region L/N=4.3 is generally

```
procedure GenSAT(Σ,Max-tries,Max-flips,p)
    for i := 1 to Max-tries
        T := initial(Σ)   ; generate an initial truth assignment
        for j := 1 to Max-flips
            if T satisfies Σ then return T
            else Poss-flips := select(Σ,T,p)   ; select set of vars to pick from
                 V := pick(Poss-flips)   ; pick one
                 T := T with V's truth assignment flipped
        end
    end;
return "no satisfying assignment found"

function select(Σ,T,p)
    if Random[0,1) < p
        then    all variables in unsatisfied clauses
        else    hclimb(Σ,T)   ; compute "best" local neighbours
```

Figure 1: The GENSAT Procedure

considered to be a good source of hard SAT problems and has been the focus of much recent experimental effort, e.g. [11, 4, 15]. The hardest SAT problems for complete procedures are, however, found at lower values of L/N [7]. This effect has not yet been found for incomplete procedures such as GSAT [6]. In this paper, we therefore test problems at L/N=4.3. To help reduce the effect of random fluctuations in problem difficulty, each experiment at a given number of variables uses the same set of 1000 satisfiable problems. As GENSAT variants typically do not determine unsatisfiability, unsatisfiable formulas were filtered out by the Davis-Putnam procedure. Since results on random problems may not be indicative of algorithm behaviour on real and structured problems, we also consider performance on non-random problems in §7.

4 Greediness and Randomness

To explore the importance of greediness in hill-climbing and randomness in picking between variables, we introduced in [4] several different variants of GSAT including CSAT, TSAT, DSAT and HSAT. CSAT is identical to GSAT except *hclimb* is more cautious, returning all variables giving an increase in score (not just the greatest increase), or if there are none, all variables giving no decrease, or otherwise all variables. TSAT is identical to GSAT except *hclimb* is timid, returning those variables that least increase the score, or if there are none, all variables giving no decrease, or otherwise all variables. For random problems, CSAT and TSAT gave very similar performance to GSAT, suggesting that greediness is not crucial to GSAT's success. DSAT is identical to GSAT except *pick* is not random but deterministic and fair. DSAT performed better than GSAT on random problems, suggesting that randomness in picking between variables is also not crucial. HSAT is identical to GSAT except *pick* uses a deterministic tabu-like restriction to pick a variable which was last flipped the longest time ago. HSAT gave the best performance of all the variants introduced in [4]. To repeat this investigation for hill-climbing procedures with random walk, we introduce four new procedures: CRSAT, TRSAT, DRSAT and HRSAT. These procedures are instances of GENSAT in which with

Table 1: The importance of random walk

Problem	Procedure	Optimal Max-flips	Total flips used median	mean	s.d.	worst case
Random 3-SAT N= 50, L/N= 4.3	GSAT	92	398	905	1,560	21,300
	GRSAT	90	349	884	2,000	48,600
	CSAT	75	402	895	1,590	23,200
	CRSAT	150	378	845	1,360	16,000
	TSAT	111	404	971	1,700	24,000
	TRSAT	205	381	815	1,230	13,000
	DSAT	60	220	482	721	5,380
	DRSAT	125	246	542	858	8,980
	HSAT	64	148	301	507	7,170
	HRSAT	195	181	413	706	9,820
Random 3-SAT N=70, L/N= 4.3	GSAT	165	1,270	2,960	5,060	69,300
	GRSAT	280	1,170	2,630	4,130	36,400
	CSAT	190	1,230	3,040	4,830	47,800
	CRSAT	378	1,100	2,430	3,610	31,700
	TSAT	213	1,370	3,390	5,790	58,500
	TRSAT	434	973	2,350	4,110	55,300
	DSAT	132	607	1,420	2,140	17,100
	DRSAT	238	636	1,570	3,020	39,100
	HSAT	116	332	743	1,160	15,100
	HRSAT	217	502	1,020	1,690	20,700
Random 3-SAT N=100, L/N= 4.3	GSAT	342	5,000	13,200	22,800	265,000
	GRSAT	719	4,220	10,300	18,200	171,000
	CSAT	544	5,330	12,600	19,700	165,000
	CRSAT	950	3,370	8,090	13,900	164,000
	TSAT	494	4,830	12,500	23,100	267,000
	TRSAT	690	3,320	8,510	17,000	329,000
	DSAT	238	2,130	5,620	9,780	129,000
	DRSAT	420	1,930	5,680	13,300	174,000
	HSAT	217	989	2,420	4,580	52,900
	HRSAT	814	1,160	2,880	6,320	134,000

probability p we perform a random walk step, and with probability $1 - p$ we use the CSAT, TSAT, DSAT and HSAT hill-climbing strategies.

Our experiments use 1000 randomly generated satisfiable 3-SAT problems at L/N= 4.3, for N= 50, 70, and 100. Since different values of p have been used previously [14, 15], we decided arbitrarily to set $p = 0.2$ for all experiments in this section. In §5.2 we investigate in detail how p should be set for some variants of GENSAT. The best performance from the best variant tested here was observed at $p = 0.2$, and it seemed a large enough value to have a significant effect on every procedure we tested. For each procedure and problem size, we determined the value of Max-flips which minimizes mean total number of flips. All our results are reported for this optimal value of Max-flips. We investigate the effect of varying Max-flips further in §5.1. The total number of flips used is a measure of computational resources.

It is very important to examine the distribution of expense, as well as broad statistical measures such as mean and median. In Table 1 we report four measures: median, mean, standard deviation,[2] and worst case. Worst case performance seems to be between 50 and 100 times the median. As problem sizes increase this factor may increase. If so, encouraging but incomplete results on very large problems such as [15], may be less good than they seem.

Table 1 compares performance of all the above mentioned variants of GENSAT, with and without random walk. As shown in [4], on these problems CSAT and TSAT are as good as GSAT, DSAT is better than all three, and HSAT is somewhat better again.

We now compare GSAT with GRSAT. For small N, there is little difference in performance. However, by N=100, GRSAT has started to outperform GSAT. This confirms suggestions that GRSAT is better than GSAT on random problems [15]. We will see later that even better performance can be achieved with a larger value of p.

Next, we consider the importance of greediness in hill-climbing, Although CRSAT hill-climbs if possible, it does not try to do so as fast as possible. Interestingly, while CSAT performs very similarly to GSAT, CRSAT outperforms both GSAT and GRSAT. Caution seems to be paying off in this case. After a downwards flip caused by a random walk step, greedy hill-climbing will often immediately flip back the same variable. By comparison, CRSAT can flip any of the variables now offered which increase the score. CRSAT may therefore reach a new part of the search space. This suggestion is tentative since the improved performance of CRSAT may not hold for optimal values of p. TRSAT's timid hill-climbing gives similar performance to CRSAT. As with cautious hill-climbing, this may be because random walk steps are often not immediately undone but move us instead to a new part of the search space. We can conclude that for random problems greediness is not crucial to the success of hill-climbing with random walk. Indeed, more cautious hill-climbing can be more effective.

Finally, we consider the importance of randomness in picking between variables to flip by testing the procedures DRSAT and HRSAT. At all problem sizes, DRSAT outperforms all variants of GENSAT which use random picking. This suggests that deterministic picking is preferable to random picking. The tabu-like picking used by HRSAT gives the best performance of all random walk procedures tested here. The optimal performance observed at each N for DRSAT and HRSAT is actually slightly worse than that found using the same algorithms without random walk. However, in the next section, we show that adding random walk to HSAT does reduce its sensitivity to choice of Max-flips. We conclude that for these problems, randomness in picking is not crucial to the success of hill-climbing with random walk, and that random walk can be helpful if incorporated with DSAT and HSAT.

5 Parameter Settings

In §4, HRSAT gave the best performance. In the rest of the paper, we focus just on variants of HSAT and GSAT with and without random walk. By restricting attention to these two procedures, we can run more comprehensive experiments.

5.1 Max-flips

Increasing Max-flips increases the probability of success on a given try, but can decrease the probability of success in a given run time. For variants of GENSAT without random walk, we observed a sharply defined optimal value of Max-flips [4]. If too small a value of Max-flip is used, there is a vanishing chance of finding a solution on any given try. If too large a value is used, the later flips of most tries are wasted work.

[2] S.d. here represents sample standard deviation, not standard error of the mean.

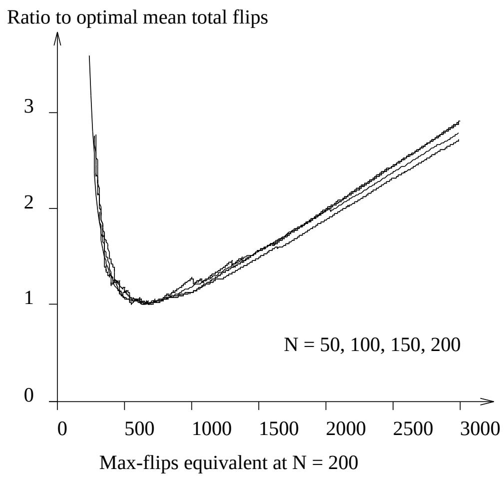

Figure 2: Scaling of performance

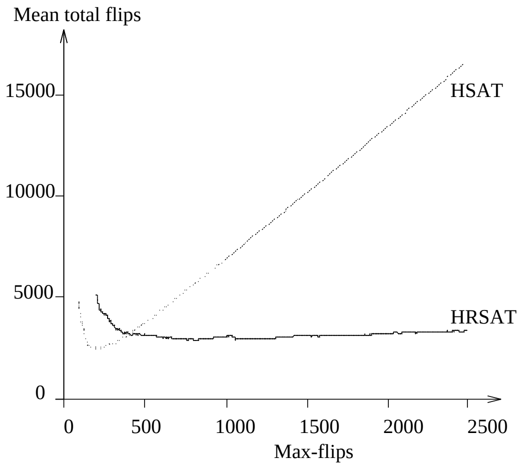

Figure 3: HRSAT vs HSAT, varying Max-flips

For HSAT and random 3-SAT problems, we found that the optimum value for Max-flips increases more than linearly with N, and in [4] we speculated that the optimal value of Max-flips varies roughly in proportion to N^2. Considerable further experimentation has allowed us to refine this conjecture. To compare performance at different problem sizes, we rescaled our data so that for each value of N, the minimum was the same. This is because we do not yet have a clear idea of how exactly optimal performance scales with problem size [4]. Having done this, performance seems to be highly predictable with varying problem size. For HSAT, we observe that optimal Max-flips scales as $N^{1.65}$. Figure 2 shows number of flips used by HSAT for each value of Max-flips for problems from 50 to 200 variables. The x-axis now represents Max-flips scaled as $N^{1.65}$. For convenience, we label the x-axis by the equivalent number of flips at N = 200. That is, the x-ordinate m in the N-variable plot represents $m \cdot N^{1.65}/200^{1.65}$. Performance is remarkably similar at different problem sizes under this scaling. This adds to a number of very simple scaling laws observed in many features of search for both complete and hill-climbing methods applied to random problems [5, 8, 10].

Figure 2 shows that for HSAT it is critical to set the value of Max-flips close to its optimal value, and to vary Max-flips as problem size changes. These two necessities represent a significant drawback to the use of GENSAT. Fortunately, adding random walk greatly reduces the sensitivity of these hill-climbing procedures to the value of Max-flips. For HRSAT with $p = 0.2$ the minimum in the total number of flips used is slightly larger than that found with HSAT. However, the setting of Max-flips for HRSAT is much less critical than for HSAT. In Figure 3, the total number of flips used is plotted against Max-flips for these two procedures, for 100 variable problems. Although the minima take similar values, the setting of Max-flips has little effect on HRSAT's performance above about 400 flips. The minimum seen here is a mean number of total flips of 2,880 at Max-flips = 814, compared to a mean of 3,340 at Max-flips=2500. This contrasts dramatically with the significant detrimental effect that increasing Max-flips above its optimal value has on the performance of HSAT.

We have seen very similar behaviour for many different variants of GENSAT with random walk, including GRSAT. It seems that near-optimal performance is obtained for a wide range of values of Max-flips. This is one of the major computational advantages of adding random walk to hill-climbing procedures. Unfortunately, the insensitivity to the value of Max-flips makes it very hard to observe a scaling law. It is difficult to get even an approximation to the optimal value of Max-flips at a given problem size.

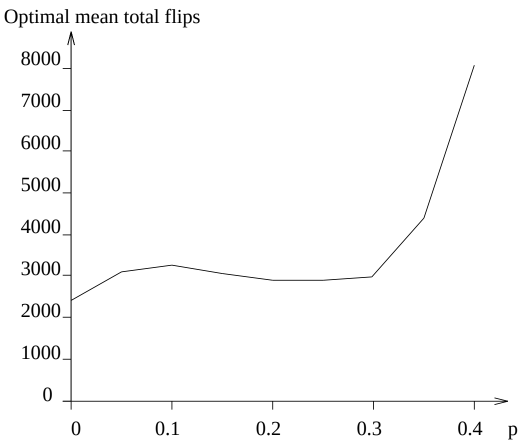

Figure 4: HRSAT, varying p

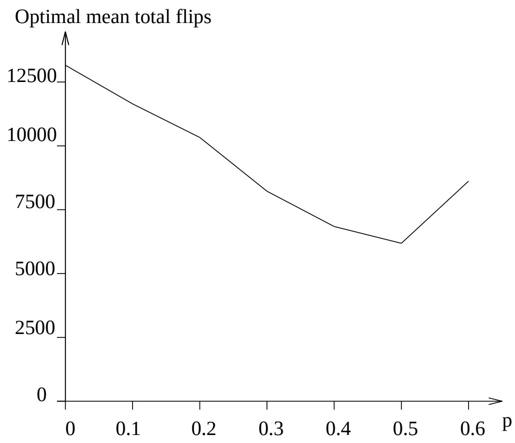

Figure 5: GRSAT, varying p

5.2 Walk Probability

The second crucial parameter for GENSAT is the walk probability p. Intuitively, we expect very small values of p to give behaviour only slightly different from that of GENSAT without random walk, and very large values to give very poor behaviour as too many random walk flips will inhibit hill-climbing to a solution. The optimal value of p should lie between these two extremes. Unfortunately as p varies, the optimal value of Max-flips also varies. Thus for each value of p considered here, the total number of flips used is that found with the optimal value for Max-flips.

Figure 4 shows how varying p affects the optimal performance of HRSAT on 100 variable problems, for p from 0 to 0.4 in steps of 0.05. There is a increase in flips used from $p = 0$ (which is simply HSAT) to $p = 0.1$. For small non-zero values of p, random walk appears to interfere with search without having significant beneficial effects. For larger values of p, the total number of flips decreases, reaching a broad minimum between $p = 0.2$ and $p = 0.3$, the best value being at $p = 0.2$ (see Table 1 for details at $p = 0.2$). In fact, this minimum is slightly worse than the performance of HSAT (again see Table 1). Nevertheless, increasing the random walk probability can improve performance. Furthermore, as described in §5.1 there is much greater freedom in the choice of Max-flips in this region of p. However, $p \approx 0.3$ seems to be close to a critical value. For larger values of p performance degrades very significantly. Care must therefore be taken in choosing p to avoid poor performance while gaining the benefits of the random walk. Although this graph shows only approximately 30% variation in performance from $p = 0$ to $p = 0.3$, more significant variation may occur at larger values of N. We were, however, restricted to N=100 in this experiment because of the expense of calculating the optimal performance of HRSAT at each point.

We repeated this experiment for GRSAT. Figure 5 shows the optimal mean total flips of GRSAT with p varied from 0 to 0.6 in steps of 0.1 on 100 variable problems. For $p = 0$ (equivalent to GSAT) the optimal performance of 13,200 flips compares unfavourably with that of 2,400 for HSAT. We do not see an initial decline in performance as we did with HRSAT. Larger values of p seemed best for GRSAT, with the best being $p = 0.5$ giving a mean of 6,140 flips. Remarkably, performance varied less than 10% when we varied Max-flips from 2,400 to 25,000, more than a factor of 10. The value of $p = 0.5$ is larger than the value $p = 0.35$ originally suggested by Selman and Kautz [14] for GSAT with random walk, but the same as they used with Cohen in a later report [15].

We conclude the random walk option improves GSAT more than HSAT for 100 variable problems, although the best settings of GRSAT still result in worse performance than HSAT or HRSAT. In addition, the choice of the walk probability p must be made in a range of approximately 0.1 for near-optimal performance, and these ranges are quite different for HRSAT and GRSAT.

To investigate further our observations in §4 about cautious hill-climbing with random walk, we varied p for a cautious variant of HRSAT, that is CHRSAT which incorporates both HRSAT's method of picking variables with CSAT's cautious hill-climbing. At 100 variables, the best value of p seemed again to be 0.2, which at Max-flips = 2,040 gave a mean total flips of 2,440, about 15% fewer than used by HRSAT at its optimal p. As with HRSAT, a very wide range of values for Max-flips give near-optimal performance. As with HRSAT, the optimum is slightly greater than the value of 2,310 flips observed for CHSAT without walk. This result supports our earlier suggestion that with random walk, cautious hill-climbing may be preferable to greedy hill-climbing.

5.3 Random Walk vs. Hill-climbing

The algorithm presented in Figure 1 allows a random walk step on any flip with a certain probability. However, if hill-climbing is possible, it may be best to flip an unsatisfied variable which increases the score as opposed to an unsatisfied variable picked at random which fails to increase (or worse still, decreases) the score. To investigate whether this is so, we implemented a variant of GENSAT in which a walk flip is only allowed if hill-climbing is not possible.

We used the cautious variant of HRSAT introduced above, CHRSAT. We varied p from 0 to 1 and at each value found the optimal value of Max-flips for 100 variable 3-SAT problems. At $p = 0$, where CHRSAT is equivalent to CHSAT, Max-flips of 200 gave a mean total flips of 2,310. As with HRSAT in §5.1, behaviour quickly declined as we increased p from 0. At $p = 0.15$ the mean total flips was 3,610. Unlike HRSAT, performance changed surprisingly little as we increased p. From $p = 0.15$ to $p = 0.75$, the best Max-flips was in the range 260-380, with a mean total flips from 3,260 to 3,910. Even setting $p = 1$, i.e. *always* make a walk flip unless an upwards flip is available, gave reasonable performance with a best mean total flips of 4,770. This is better, for example, than the best behaviour found for GRSAT for the same number of variables. On the other hand, it is worse than the best values for CHRSAT in which random walk steps can always be performed. Also, unlike earlier variants of GENSAT, we observed a sharp minimum in the total number of flips. One of the great advantages of GENSAT with random walk, the insensitivity to the setting of Max-flips, has therefore been lost.

Good behaviour at $p = 1$ suggests that we can discard plateau search, i.e. sideways moves when no improvements are possible, and still retain good performance. Earlier reports on GSAT and variants of it suggested that plateau search was crucial to performance [16, 4]. Further experiments are needed to determine if plateau search can indeed be discarded since these variants of CHRSAT may be simulating plateau search. If a random walk flip is made which decreases the number of satisfied clauses, hill-climbing may return search back to the plateau close to where it was left. We may therefore be merely simulating plateau search rather inefficiently. Our results do suggest that random walk should be allowed even when hill-climbing is possible.

6 Jump-SAT

We have shown that the optimal probability of making a walk step can be quite large, and that walk steps should be allowed even when hill-climbing is possible. The probability of making two walk steps in a row (that is, p^2) is therefore appreciable. This suggests that it is important to perform a lot of random walk compared to hill-climbing, and to allow random walk to move large distances around the search space. To determine if performance can be improved by making larger walk steps, we implemented a new algorithm called "JumpSAT". This is similar to GRSAT, except that instead of flipping just *one* variable in an unsatisfied clause with probability p, we flip one variable from *each* unsatisfied clause. After one such jump, all clauses that were previously unsatisfied become satisfied. Although some previously satisfied clauses become unsatisfied, subsequent hill-climbing and jumps may satisfy these clauses.

Following the results of §5.3 we first allowed the Jump step to be allowed even when hill-climbing was possible. Compared to GRSAT a very small probability was required. On 100 variable random 3-SAT problems at L/N= 4.3, we observed best performance of a mean 9790 total flips at $p = 0.03$. For 70 variable problems, JumpSAT only marginally outperformed GSAT with this setting of p, and for 50 variable problems it was slightly worse than GSAT. We also tested a version of JumpSAT in which Jump steps are prohibited when hill-climbing is possible. At N=100 and $p = 0.1$, we observed an optimal mean total flips of 9000 with Max-flips = 470. At this value of p, JumpSAT also outperformed GRSAT for 50 and 70 variable problems. Furthermore, at any value of p we tested up to 0.5, better performance was obtained than with GSAT. Thus both variants of JumpSAT are much better than GSAT but not so good as GRSAT at $p = 0.5$. However, when jumps are allowed on all flips, the setting of p and Max-flips are much more critical than GRSAT.

We have shown that JumpSAT outperforms GSAT, although we have not yet shown it to be better than GRSAT. Further experimentation with JumpSAT and variants of it may produce even better results.

7 Non Random Problems

In this section, we describe the performance of GENSAT on several structured problem classes. Since some of these problems appear to be very hard for GENSAT, we were unable to perform experiments at different values of p. Therefore, throughout this section, we used the values of $p = 0.5$ for GRSAT and 0.2 for HRSAT suggested by §5.2. Similar comments apply to Max-flips, and we chose to set Max-flips = 1000N unless otherwise stated. This very large value is sometimes necessary. Throughout, we compare performance with complete procedures like Davis-Putnam. Finally, in §7.5, we examine the effect of model density on the two types of procedure.

7.1 n-queens problem

The n-queens problem is a standard problem for constraint satisfaction algorithms. Our encoding into SAT uses n^2 variables, one for each square on a chessboard. GENSAT without walk has been shown to perform very well on this encoding [16, 4]. To determine how random walk affects performance, we tested GRSAT and HRSAT on the 6-, 8-, and 16-queens problems averaging over 1000 runs. In every run both procedures solved these problems on the first try. For comparison, we give results for GSAT at Max-flips = 5N. Although the random walk option resulted in worse mean performance, it was still good, and did avoid getting stuck in local maxima which can happen to GSAT on the n-queens problem. The Davis-Putnam procedure finds this encoding of the n-queens problem quite hard. Between 8-queens and 16-queens, the amount of search for the Davis-Putnam procedure increases by a factor of more than 40.

Table 2: Random walk and the n-queens problem

Problem	Procedure	Total flips used			
		median	mean	s.d.	worst case
6-queens	GSAT	198	271	267	1760
	GRSAT	200	281	279	2400
	HRSAT	198	330	379	2590
8-queens	GSAT	71	141	170	1960
	GRSAT	134	179	147	1040
	HRSAT	84	161	214	1940
16-queens	GSAT	207	288	251	1800
	GRSAT	466	594	432	4480
	HRSAT	238	290	165	1200

7.2 Quasigroups

Several open problems in finite mathematics concerning the existence of quasigroups have recently been solved by encoding into SAT [17]. A quasigroup is described by a Latin square, a multiplication table with a simple closure property. These results are of practical interest as certain results in design theory reduce to questions about the existence of quasigroups with appropriate properties. We tested behaviour on the problems QG1.n, QG2.n and QG3.n (the names are taken from [17], n is the size of the quasigroup). These problems encode into SAT using n^3 variables.

We did not find GENSAT well suited to these problems, with or without random walk. For example, HSAT required 311 tries to solve even a simple problem like QG1.5 with Max-flips = 10N= 1250. GRSAT did manage to solve it on its first try, but required 96,764 flips to do so. HRSAT failed to solve it in 10 tries. GRSAT failed to solve QG1.7 or QG2.7 in 6 tries. GRSAT and HRSAT failed to solve QG3.8 in 10 tries. Recall that all these failures were at Max-flips = 1000N. HSAT also failed to solve QG3.8, in 1000 tries at Max-flips = 10N. For comparison, several open problems with between 343 and 3375 variables have been solved using a variety of procedures based on constraint satisfaction, resolution and Davis-Putnam [17].

7.3 Factorization

Jong and Spears have proposed a novel class of problems based upon an encoding of factorization into SAT [9]. The encoding constructs a boolean circuit which multiplies two binary words giving as an output the binary encoding of n. If this circuit has a model then the inputs give two factors of n. To ensure that all problems have a solution, we allow 1 as a factor of any number. We tested GRSAT on 8-bit factorization problems, the encodings of which have 184 variables. GRSAT factored all the numbers from 1 to 100 on the first try. The mean number of flips was 45,400, with a standard deviation of 39,800, and worst case of 181,066 (pushing the limit of 1000N flips). This is a great improvement over GENSAT without walk, because HSAT, which seems to be the best such variant [4], failed on all of these problems, either given 5 tries at Max-flips = 1000N or 50 tries at Max-flips = 25N. GRSAT outperformed HRSAT, which solved only 67 problems within 5 tries. It is hard to conclude that GRSAT is better on these problems since we may have been using non-optimal values of p. Although GRSAT solved these problems, it does not compare well with a complete procedure such as Davis-Putnam, which never required more than 100 branches to find a solution.

7.4 Zebra

Our final example is a logical puzzle called the zebra problem. This has been used as a benchmark problem in the constraint satisfaction community, for example in [3, 13]. The SAT encoding uses 205 variables and 2975 clauses. Results with GENSAT suggest that random walk is helpful. HSAT failed to solve the problem in 2,500 tries at Max-flips = 25N. HRSAT solved it 10 times in 100 tries, taking on average 63,800 flips. GRSAT solved it 23 times in 100 tries, in on average 56,300 flips. By comparison, the zebra problem is not hard for Davis-Putnam, which can take advantage of the many constraints. Our implementation needed just 114 branches, searching to a maximum depth of 13 splits, performing 1663 unit propagations and 183 pure deletions.

7.5 Model density

GENSAT with walk performs well on the n-queens problem but poorly on the other non-random problems tested here (further examples of non-random problems where GRSAT performs well are given in [15]). By comparison, Davis-Putnam behaves poorly on the n-queens problem but comparatively well on the other non-random problems. This difference in behaviour seems to be related to the model density. For example, note GENSAT's poor performance on the 6-queens problem in Table 2. This is the largest queens problem with a unique solution up to symmetry. It thus has a relatively small model density. For larger n, the n-queens problem rapidly becomes easy as the problem becomes more underconstrained, and the number of models increases rapidly. Although there are many models, it easy for Davis-Putnam to make an early mistake that leads to much irrelevant search. A similar "early mistake" phenomenon has been seen in random problems in the satisfiable region [7].

By comparison, there are several reasons why GENSAT finds the quasigroup problems so hard. First, the number of models for the quasigroup problems is very small compared to the number of truth assignments. Second, the encoding only indirectly preserves many of the natural constraints (*e.g.* that the quasigroup is closed under multiplication). Third, the size of a flip may be too small since it changes just one entry in the multiplication table. If flips permuted two entries, the multiplication table would remain closed. Because of this small flip size, the search space is much larger than it need be. Much of search is spent exploring truth assignments which do not represent quasigroups. Finally, hill-climbing procedures do not propagate constraints well, unlike conventional backtracking procedures. The encoding includes many binary clauses, so unit propagation does much of the work in Davis-Putnam. Hill-climbing searches many truth assignments which unit propagation would rule out as trivially incompatible.

Similar comments apply to factorization problems. Hill-climbing procedures seem to find factoring almost uniformly difficult, no matter how many factors a number has. Indeed, even even numbers are hard to factor. This is not too surprising given the small number of models. If a number has m factors, then the boolean circuit has just $O(m)$ models and this is small compared to the 2^N possible truth assignments. Similarly, the zebra problem has a *unique* model amongst 2^{205} truth assignments. Using hill-climbing, it is hard to find the model from the many near models. However, unit propagation again saves much search for Davis-Putnam making the problem easy.

It is interesting to compare this situation with hard random k-SAT problems, where GRSAT seems to be better than Davis-Putnam [15]. Whilst the model density tends to zero as N increases, the expected number of models *increases* exponentially. Random 3-SAT problems at N=100 and L/N=4.3 have on average more than 146,000 models, but they are not systematically constrained and thus can be hard for Davis-Putnam. The quasigroup, factorization, and zebra problems have orders of magnitude fewer models despite containing more variables. We speculate that the number of models in relation to problem size may be crucial to the performance of hill-climbing procedures.

8 Related Work

Papadimitriou [12] proposed a simple random walk algorithm for 2-SAT which repeatedly flips unsatisfied variables chosen at random. For satisfiable problems, this algorithm finds a model in $O(\mathrm{N}^2)$ flips with probability approaching 1 as N increases.

Selman and Kautz proposed adding random walk to GSAT in [14]. With Cohen, they showed that this combination improves the performance of GSAT both on hard random problems and certain structured problems like boolean circuit synthesis [15]. On hard random problems, we have shown here that the optimal performance of GSAT with random walk is still worse than that of the algorithm HSAT introduced in [4].

WSAT, introduced in [15], also focuses search on unsatisfied variables. WSAT chooses an unsatisfied clause at random, and then picks an unsatisfied variable to flip from this clause using either a greedy or a random heuristic. With a random picking heuristic, WSAT is simply Papadimitriou's random walk algorithm with restarts. On some circuit synthesis and diagnosis problems, WSAT outperformed GSAT with random walk [15]. These results are, however, too preliminary to determine if WSAT consistently outperforms GSAT with random walk, or the better variants like HRSAT introduced here. Note that WSAT is a simple variant of GENSAT in which *hclimb* returns the variables in an unsatisfied clause, and *pick* chooses between then either randomly or greedily.

As with random walk, simulated annealing modifies conventional hill-climbing by allowing occasional downward flips. Comparisons between GSAT with random walk and simulated annealing are given in [1, 15]. The first study shows similar performance for the two algorithms, whilst the second shows GSAT with random walk performing better than simulated annealing. Note that, unlike random walk, simulated annealing does not focus downward flips on unsatisfied variables.

9 Conclusions

We have introduced several new procedures for satisfiability which use a random walk mechanism to focus search on the unsatisfied variables. Some of these procedures are able to outperform GSAT with random walk, currently one of the best hill-climbing procedures for satisfiability [14]. For random problems, we have shown that the greatest benefit of adding random walk is not the improvement in optimal performance but the reduction in the sensitivity to input parameters. Indeed, on hard random problems, the optimal performance of GSAT with random walk appears to remain worse than that of a hill-climbing procedure like HSAT without random walk. We also observed a simple scaling law in performance on random problems. For non-random problems, although random walk can improve performance, many problems remain difficult. We speculate this is because the number of models is small and local search is unable to take advantage of the constraints which greatly reduce the difficulty of such problems for conventional procedures like Davis-Putnam. Unless constraint propagation techniques can be profitably combined with local search, such non-random problems are likely to remain hard for hill-climbing procedures.

Acknowledgements

The second author was supported by a HCM Postdoctoral Fellowship, and the first author partly by a SERC Postdoctoral Fellowship and partly by his wife, Judith Underwood. We thank Alan Bundy and the members of the DReaM Group for their constructive comments and many CPU cycles donated to these experiments from SERC grant GR/H/23610. We also thank Bob Constable, Pierre Lescanne, Fausto Giunchiglia, the Eureca group at INRIA-Lorraine, the Department of Computer Science at Cornell University, and the MRG group at Trento for additional CPU cycles; Mark Stickel for code to generate quasigroup problems; and our referees for helpful comments.

References

[1] A. Beringer, G. Aschemann, H. H. Hoos, M. Metzger, and A. Weiss. GSAT versus simulated annealing. *11th European Conference on AI*, pages 130–134, 1994.

[2] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. *Proc. 12th IJCAI*, pages 331–337. IJCAI, 1991.

[3] R. Dechter. Enhancement schemes for constraint processing: backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[4] I. P. Gent and T. Walsh. Towards an Understanding of Hill-climbing Procedures for SAT. *Proc. 11th National Conference on AI*, pages 28–33. AAAI Press, 1993.

[5] I. P. Gent and T. Walsh. An Empirical Analysis of Search in GSAT. *Journal of Artificial Intelligence Research*, 1:47–59, September 1993.

[6] I.P. Gent and T. Walsh. The hardest random SAT problems. In B. Nebel and L. Dreschler-Fischer, eds. *KI-94: Advances in AI.* pp 355-366, Springer-Verlag, 1994.

[7] I. P. Gent and T. Walsh. Easy Problems are Sometimes Hard. *Artificial Intelligence*, 70:335–345, 1994.

[8] I. P. Gent and T. Walsh. The Satisfiability Constraint Gap. To appear in *Artificial Intelligence.*

[9] K.A. De Jong and W.M. Spears. Using Genetic Algorithms to Solve NP-Complete Problems. *3rd Int. Conference on Genetic Algorithms.* Morgan Kaufmann, 1989.

[10] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, May 27 1994.

[11] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. *Proc. 10th National Conference on AI.* AAAI Press, 1992.

[12] C.H. Papadimitriou. On selecting a satisfying truth assignment. *Proc. Conference on the Foundations of Computer Science*, pages 163–169, 1991.

[13] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.

[14] B. Selman and H. Kautz. Domain-independent extensions to GSAT: solving large structured satisfiability problems. *Proc. 13th IJCAI*, pages 290-295, 1993.

[15] B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. *Proc. 12th National Conference on AI*, pages 337–343. AAAI, 1994.

[16] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. *Proc. 10th National Conference on AI.* AAAI Press, 1992.

[17] J. Slaney. The Crisis in Finite Mathematics: Automated Reasoning as Curse and Cure. In A. Bundy, editor, *12th Conference on Automated Deduction*, pages 1–13. Springer Verlag, 1994.