The Enigma of SAT Hill-climbing Procedures *

Ian P. Gent & Toby Walsh
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
I.P.Gent@edinburgh.ac.uk
T.Walsh@edinburgh.ac.uk[†]

Abstract

In this paper, we investigate a family of hill-climbing procedures related to GSAT, a greedy random hill-climbing procedure for satisfiability. These procedures are able to solve large and difficult satisfiability problems beyond the range of conventional procedures like Davis-Putnam. We explore the rôle of greediness, randomness and hill-climbing in the effectiveness of these procedures. We show that neither greediness nor randomness is crucial to GSAT's performance, and that hill-climbing's importance is limited to a short initial phase of search. In addition, we observe some remarkable and possibly universal features of their search for a satisfying truth assignment.

*The title is an allusion to the procedure developed by Alan Turing for breaking the Enigma code in World War II. This procedure began with a guess and successively refined it according to the contradictions it generated. This bears a remarkable similarity to GSAT, the procedure analysed in this paper.
Due to the discovery of a bug in some of our code, the data in this paper is slightly different to that contained in a preliminary version. Fortunately the differences are small and our conclusions are unaffected.

†We thank Gordon Reid and Bart Selman for their programming assistance, and the members of the Mathematical Reasoning Group for their constructive comments and CPU cycles.

1 Introduction

Many problems in AI are NP-hard and are thus, in general, intractable. A solution to this intractability is to give up completeness; that is, instead of an algorithm which is guaranteed to return an answer, we provide a tractable procedure which will often return an answer but may sometimes terminate before an answer is found. One such procedure is GSAT [Selman *et al* 92], a random greedy hill-climbing procedure for propositional satisfiability, or SAT (that is, the problem of deciding if there is an assignment for the variables in a propositional formula that makes the formula true). SAT is an interesting problem since it was the first computational task shown to be NP-hard [Cook 71]. It is also useful since many AI problems can be encoded quite naturally in SAT (*eg.* planning [Kautz & Selman 92], constraint satisfaction, diagnosis and vision interpretation [Reiter & Mackworth 89]). In addition, satisfiability is closely related to theorem proving; refutation procedures call upon the fact that φ is derivable from Σ iff $\Sigma \cup \neg\varphi$ is not satisfiable. Unfortunately, conventional algorithms for SAT like Davis-Putnam [Davis & Putnam 60] can take a time exponential in the length of the formula to run. GSAT, however, seems to scale much better; it is able to solve large and difficult SAT problems which are beyond the reach of Davis-Putnam [Selman *et al* 92].

GSAT deals with the satisfiability of conjunctive normal form (CNF) formulae; a formula, Σ is in CNF iff it is a conjunction of clauses, where a clause is a disjunction of literals. GSAT is, as we have said, a random greedy hill-climbing procedure. GSAT starts with a randomly generated truth assignment, and hill-climbs by changing the variable assignment which gives the largest increase in the number of clauses satisfied. GSAT is greedy since it flips the variable which gives the largest increase. GSAT is random since it starts from a random truth assignment and since, given the choice between several equally "good" flips, it picks one at random. GSAT hill-climbs on the number of clauses satisfied (which we will call the "score" from now on); one very important feature of this hill-climbing is the use of sideways flips – if there exists no flip which increases the score, then a variable is flipped which does not change the score.

```
procedure GSAT(Σ)
    for i := 1 to Max-tries
        T := random truth assignment
        for j := 1 to Max-flips
            if T satisfies Σ then return T
            else Poss-flips := set of vars which increase satisfiability most
                 V := a random element of Poss-flips
                 T := T with V's truth assignment flipped
        end
    end
    return "no satisfying assignment found"
```

The aim of this paper is to try to explain in more detail why this simple procedure works so effectively. In particular, we will focus on three questions. Is greediness important? Is randomness important? Is hill-climbing important? Our answers will be based on observations about the nature of the search space and on several variants of GSAT, some of which show improved performance on certain classes of problems. Our answers suggest that neither greediness nor randomness is crucial to GSAT's performance, and that hill-climbing's importance is limited to a short initial phase of search. We will also conjecture the existence of several universal features of the search.

2 GenSAT

To investigate these questions, we introduce a generalisation of GSAT, called "GenSAT", in which different methods of hill-climbing can be implemented.

procedure GenSAT(Σ)
    for i := 1 **to** Max-tries
        T := *initial*(Σ) ; initial truth assignment
        for j := 1 **to** Max-flips
            if T satisfies Σ **then return** T
            else Poss-flips := *hill-climb*(Σ, *T*)
                    V := *pick*(Poss-flips)
                    T := T with V's truth assignment flipped
        end
    end
    return "no satisfying assignment found"

GSAT is an instance of GenSAT where *hill-climb* returns those variables which increase the score most by being flipped, while *pick* chooses a random member of the set given to it, and *initial* simply sets each variable independently to true or false with equal probability. Note that hill-climbing procedures like GSAT's need to be implemented with some care. Naively, one simply iterates through the variables, and for each one calculates the number of clauses satisfied if that variable is flipped. For N variables and L clauses, the work involved is O(NL). However, if we store for each variable the change in score that flipping it would make, it is possible to update this information after each flip just by examining each clause that the flipped variable appears in.[1] In random problems where the ratio between clauses and variables is held constant, the number of clauses the average variable appears in is also constant as N varies. Even for small problems such as N=50, the resulting speedup is by orders of magnitude. A similar technique can be applied to hill-climbing since we need examine all variables only if the score of all variables that were previously best has decreased.

[1] We thank Bart Selman for pointing this out to us.

Note that we can make GenSAT a *complete* decision procedure for satisfiability by having *initial* generate, perhaps in a random order, all 2^N possible truth assignments.

3 Greediness and Caution

The "G" in GSAT comes from its *greediness*. To study the actual importance of greediness, we introduced CSAT, a *cautious* procedure. CSAT is an instance of GenSAT in which *pick* is as before, but hill-climbing acts not on the value of the change in score, but only on the sign of the change in score. That is, *hill-climb* returns all variables which effect a positive change in score, or if there are no such variables, all variables which make no change, or if there are none of those, all variables are returned.[2] The intuition is that since greedy hill-climbing gives at best a gross approximation to the correct variables to flip, using an approximation to greedy hill-climbing is likely to give similar performance.

To compare GSAT and CSAT, we ran each procedure on the same problem sets. We did this for two types of problems, the n-queens problem and instances of random k-SAT. The n-queens problems is to place n-queens on an $n \times n$ chessboard so that no two queens attack each other. Problems in Random k-SAT with N variables and L clauses are chosen as follows: for each clause a random subset of size k of the N variables is selected and each of these variables is made positive or negative with probability $\frac{1}{2}$; each of the L clauses are chosen independently in this way. For $k = 3$ the ratio L/N = 4.3 has been identified as being the region where the probability of satisfiability is about $\frac{1}{2}$ and where Davis-Putnam performs particularly badly [Mitchell *et al* 92]. This ratio was used in an earlier study of GSAT [Selman *et al* 92]. Because GSAT never succeeds on unsatisfiable formulas, we filtered these formulas out by passing each formula through the Davis-Putnam procedure. This approach put a strict limit on the size of random problems we could investigate, and our main studies were on 50-variable and 215 clause problems in Random 3-SAT. The results are summarised in the following table.

Problem	Procedure	Tries	Flips
Random 50 Vars	GSAT	6.08	95.2
	CSAT	5.46	98.1
Random 100 Vars	GSAT	25.0	262
	CSAT	27.5	268
6-queens	GSAT	2.29	66.7
	CSAT	2.14	72.0
8-queens	GSAT	1.16	85.4
	CSAT	1.22	101
16-queens	GSAT	1.03	253
	CSAT	1.03	274

[2]Except when there are variables which lead immediately to a solution, in which case those are returned. In practice, this exception didn't seem to affect performance.

In each case Max-flips was set at 5 times the number of variables (the n-queens problem can be written down using n^2 variables) while Max-tries was set at 999. The figures for tries are the average number of tries taken until success (the minimum being 1 of course), while the figures for flips gives the average number of flips in successful tries only. In each case 500 experiments were performed, all of which were successful.

As these results indicate, the performance of CSAT is closely comparable to that of GSAT.[3] There is, however, a tendency for CSAT to take a few more flips. We explain this by the obvious fact that CSAT climbs hills more slowly than GSAT, but it seems to do it just as surely. Cautious hill-climbing is not trapped as easily by local maxima as greedy hill-climbing. However, this did not contribute significantly to the results since local maxima occurred infrequently: less than 1% of tries encountered a local maximum.

We also implemented a variant of CSAT in which all positive moves were selected, but if there were none, all variables were chosen: that is, sideways moves were not preferred to downwards. This variant of CSAT performed much less well than CSAT itself, typically requiring Max-flips to be doubled for similar performance; a similar result was reported for GSAT without sideways moves in [Selman *et al* 92].

We have found no problem sets where CSAT performed significantly worse than GSAT. Indeed, we found no variant of GSAT that performed better than the equivalent variant of CSAT. This strongly suggests that caution is just as good as greediness for hill-climbing.

4 Randomness and Determinism

Of the three auxiliary procedures in GenSAT, GSAT uses randomness in two of them; the initial assignment selected by *initial* and the choice of variables given by *pick*. What happens if we eliminate the randomness from these procedures? We investigated eliminating this randomness from each procedure independently.

First we changed *initial* so that it would always return the same truth assignment (and on random problems this might as well be the assignment making every variable false). On 500 problems with 50 variables this variant failed to solve all the problems in 999 tries, taking about 50 tries on average. It thus seems that the choice of initial truth assignment is crucial to the performance of GSAT, and must vary from try to try. We have not investigated whether effective performance can be achieved by making these truth assignments vary in a deterministic way, and this remains an interesting research project.

To investigate the importance of randomness in *pick* we implemented two deterministic versions of it. The resulting deterministic variants of GSAT are called "DSAT". We implemented two versions, because a crucial issue seemed to be *fairness*. We say that a particular *pick* is fair if it is guaranteed to eventually pick

[3] It is interesting to note that the 6-queens problem is the hardest of the queens problems, at least comparatively. Presumably it is not coincidence that it is the largest problem that has a unique solution up to symmetry.

a variable if it is continuously in the set of variables selected by hill-climbing.[4] Random picking is certainly fair. Our fair version of *pick* collects the variables into an arbitrary but fixed cycle, and records which variable was picked last. The variable picked is then the next variable in this cycle, out of those variables selected by *hill-climb*. Our unfair version of *pick* simply returns the least variable (in a standard ordering) in the list presented to it by *hill-climb*. This fair version of DSAT is similar to SAT6.0 [Gu 92].

The unfair version of DSAT performed extremely poorly, although it is perhaps surprising that it achieves any success at all. Even with 999 tries it achieved only a 93% success rate on random problems with 50 variables, taking on average 180 tries and 17 flips to achieve those successes. The low number of flips simply indicates that unless it gets the answer almost instantly, it never will. We doubt that any unfair variant of GenSAT could produce results comparable to GSAT.

Results for the fair version of DSAT, on the same problems as before, are summarised in the following table: figures in brackets give the result as a percentage of GSAT's result on the same problem.

Problem	Procedure	Tries		Flips	
Random 50 Vars	Fair DSAT	4.69	(77%)	72.8	(77%)
Random 100 Vars	Fair DSAT	14.3	(57%)	226	(86%)
6-queens	Fair DSAT	1.10	(48%)	45.8	(69%)
8-queens	Fair DSAT	1.10	(96%)	45.7	(54%)
16-queens	Fair DSAT	1.02	(99%)	150	(59%)

The fair variant of DSAT actually gives *improved* performance over GSAT. Some of the improved performance is accounted for by the fact that deterministic *pick*, unlike random *pick*, is guaranteed not to repeat its last move unless there is no other choice: we return to this point in Section 6. Comparable results were obtained for the fair deterministic variant of CSAT: this variant of GenSAT is close to the algorithm SAT1.1 [Gu & Gu 91].

These results strongly suggest that the initial truth assignment must be allowed to vary, and that there is nothing essential about the randomness of picking in GSAT, but fairness is important.

5 Hill-climbing

The third feature of GSAT we investigate is the importance of its hill-climbing. We implemented a version of GenSAT called "RSAT" in which *hill-climb* returns every variable, and *pick* chooses one of these variables at random. RSAT therefore does no hill-climbing, and simply performs a random walk over the space of truth assignments. Theoretically we expect RSAT's performance to depend solely on

[4] This allows a variable *not* to be picked if it is presented, say, only every other time. Perhaps a stronger notion of fairness is required although the version used here is sufficient for this investigation.

the density of models; this decreases exponentially with L but is independent of N. This conjecture was confirmed experimentally: RSAT was able to solve random satisfiable 3-SAT problems for L=50 for any ratio of L/N using on average 125 flips and 5 tries; GSAT, by comparison, took only 12 flips and 1.5 tries for L=50 and the difficult L/N=4.3 region. RSAT was not able to solve 3-SAT problems in which L was much larger. We estimate that it would take $O(10^{12})$ flips on average for RSAT to solve 3-SAT problems with L=215; this is much worse than GSAT or even the Davis-Putnam procedure.

Hill-climbing is therefore important. However, on typical runs of GSAT and CSAT on random problems, 90% of all flips are sideways moves. The vast majority of GSAT's search is thus not governed by climbing. This can be graphically illustrated by plotting the score and poss-flips (the number of variables returned by *pick*) against the number of flips performed. Both score and poss-flips are plotted as a percentage of their maximum values. These graphs provide insight into the topology of the search space.

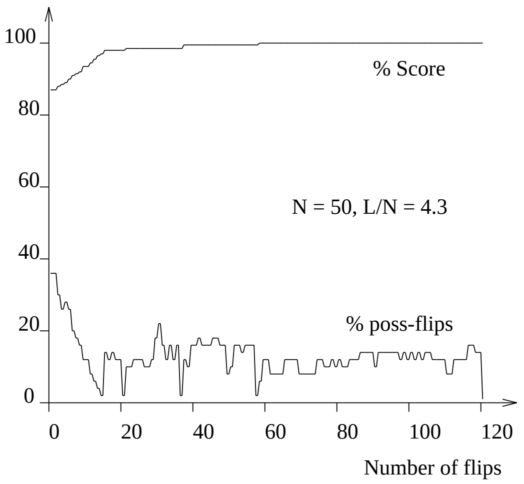

Figure 1: CSAT behaviour on one try

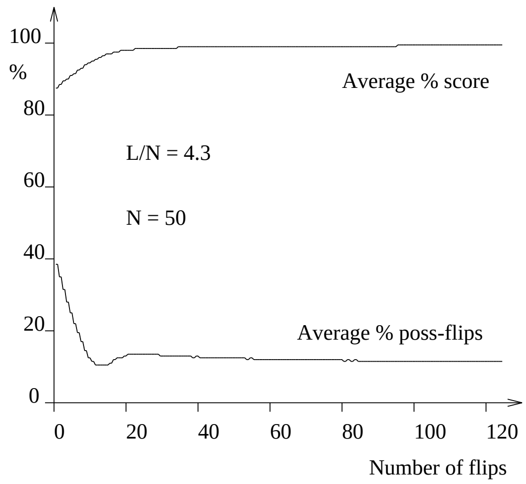

Figure 2: Average CSAT behaviour

We begin with a typical run of CSAT (figure 1) on a random 3-SAT problem. Although there is considerable variation between runs, this graph illustrates features common to all runs. The percentage score starts at 87%, which seems surprisingly high. Theoretically, however, we expect a random truth assignment in k-SAT to satisfy $\frac{2^k-1}{2^k}$ of all clauses (in this instance, $\frac{7}{8}$). For the first 16 or so flips, the score climbs rapidly. From then on, CSAT walks sideways across a stepped plateau. Note that the steps in the percentage score correspond to *integer* increases in the score and not to any discreteness of the plotting. Poss-flips, on the other hand, drops rapidly during the initial climbing phase reaching a minimum of 1 as we go onto the plateau. Once on the plateau, poss-flips is larger than this since approximately 10% of the variables move us sideways. Although the behaviour of poss-flips on the plateau is quite noisy, there is one noticeable feature: poss-flips dips sharply to one or two variables whenever we go up a step. We conjecture that, when CSAT is on the plateau and offered a chance a step up, there are only ever a few flips which increase the score. After 121 flips, a model was found.

To summarize, CSAT appears to have a short climbing phase, followed by a random walk over a stepped plateau. This picture of CSAT's behaviour is preserved well when we average over a large number of runs, and when we vary N, L and k. In addition, some remarkable new features appear which are not very apparent in the noise of a single run. For example, in figure 2 the average behaviour of CSAT over 500 runs is plotted. As predicted theoretically, the score begins at $\frac{7}{8}$. The percentage score climbs rapidly for about 20 flips and then flattens off as we mount the plateau. The steps in this curve correspond to discreteness of the plotting since the steps that occur in an individual run are smoothed out by averaging. Poss-flips, on the other hand, drops sharply during the initial climbing phase, reaching a very pronounced minimum just before CSAT steps onto the plateau. Once on the plateau, poss-flips decays slowly.

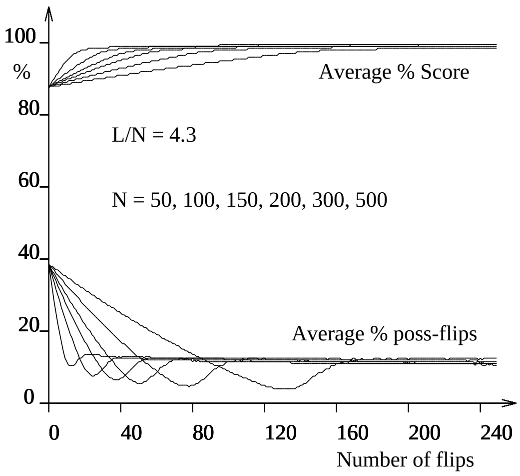

Figure 3: Average CSAT behaviour

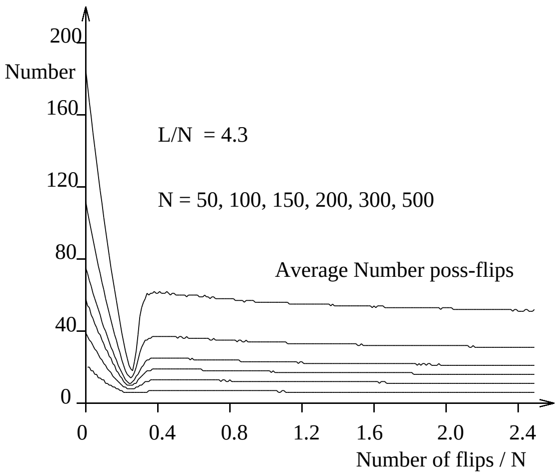

Figure 4: Average CSAT behaviour

This behaviour is duplicated for other values of N. In figure 3, N takes values between 50 and 500 whilst L/N is fixed at the difficult 4.3 ratio. As figure 4 demonstrates, the minimum value of poss-flips occurs after about 0.25N flips. This corresponds to the average point where the plateau is reached. By 0.4N flips the plateau has almost always been reached.

GSAT displays similar behaviour to CSAT with some additional and interesting features. Figure 5 is from a single run of a random problem with N=500 and the number of clauses set at 2150 so that L/N=4.3. First note that the average score increases in the same way as it did for CSAT, although slower than in figure 1 since the number of variables and clauses is ten times greater. The large number of clauses also means that some of the discreteness in the score is lost when plotted to this small scale. Perhaps the most fascinating feature is the initial behaviour of poss-flips. The graph shows three well defined wedges starting at 15, 30, and 52 flips. No such wedges were observed during runs of CSAT although the tail end of the last wedge is familiar. The sharp rise in poss-flips at 52 flips is the point where it became impossible to increase the score and only sideways moves became possible. The earlier sharp rises at 15, and 30 flips represent similar changes. The last wedge spans the region where flips typically increase the score by 1. The middle wedge spans the region where flips typically increase the score by 2, whilst in the first wedge, scores typically increase the score by 3. As with CSAT and the

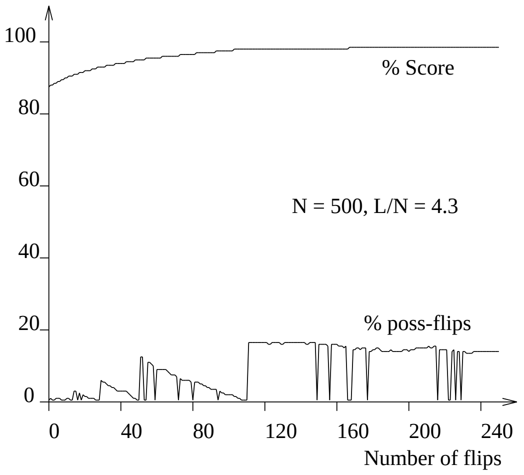

Figure 5: GSAT behaviour on one try

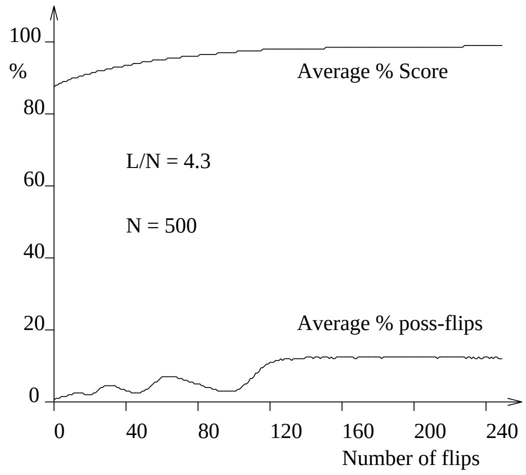

Figure 6: Average GSAT behaviour

transition from positive to zero change in the score, each change in the maximum possible increase in score is marked by a sharp increase in poss-flips. Dips in the wedges represent unusual points where individual flips could increase the score by more than the characteristic value for that region, just as the dips in poss-flips after the plateau is reached represent points where an increase in score was possible. This behaviour is observable when GSAT's behaviour is averaged over a large number of runs (figure 6). The wedges are smoothed out into curves since their starting points vary between runs. We conjecture that as N increases, the number of these wedges will increase without limit.

Both CSAT and GSAT's average behaviour scales remarkably. In figures 7 and 8, the score and poss-flips have been plotted as percentages for six values of N from 50 to 500. In both cases, the *six* graphs coincide almost exactly. The only differences occur at the minima, which become sharper with increasing N. This linear scaling is a fascinating phenomenon for which we welcome explanations.

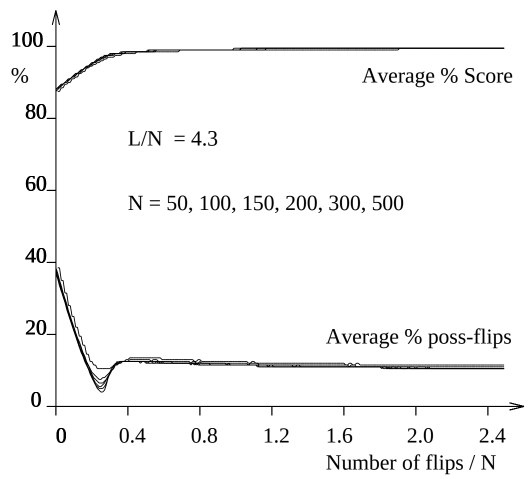

Figure 7: Average CSAT behaviour

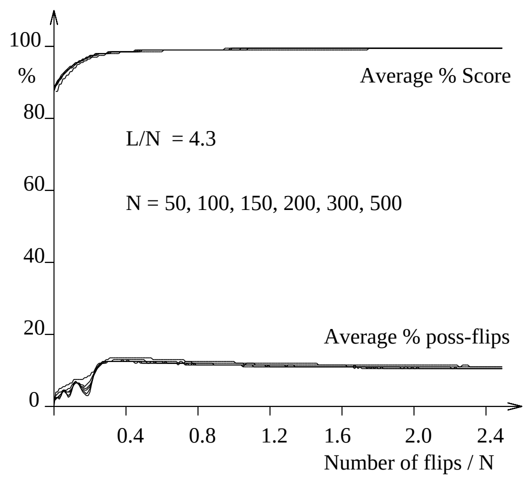

Figure 8: Average GSAT behaviour

The results we have obtained have important implications. First, note that climbing only dominates GSAT's search for about 0.3N flips and CSAT's for 0.4N flips. This is to be expected since GSAT climbs faster than CSAT. After this, search is dominated by sideways moves on the plateau and GSAT and CSAT behave identically. This part of the search is much more of a random walk than hill-climbing, and improving this is the way to better GenSAT's performance. As shown by RSAT, hill-climbing is crucial to GSAT's effectiveness: now we can state with more certainty that its contribution is to enable GSAT to attain regions of high-score very rapidly. Thereafter the problem is to find points where any climbing at all is possible. This justifies the similarity between GSAT and CSAT after 0.4N flips; neither spend much time climbing.

Another important feature of the graphs of average behaviour is that poss-flips declines very slowly once on the plateau. On fifty variable problems the average number of poss-flips was 0.09N after 10,000 flips. Assuming this scales to 500 variable problems, as our results strongly suggest, we would still be faced with an average of 45 variables to choose from after 100,000 flips. This information can be used to guide methods of improving GenSAT's search of the plateau, and possibly to suggest estimates of the expected running time of variants of GenSAT. We have only had time to investigate one possibility to date, which we describe in the next section.

In summary, both GSAT and CSAT climb consistently for the first 0.3N and 0.4N flips respectively and then perform a random walk, climbing on the rare occasions where it is possible. Furthermore, we have noted some interesting and remarkably consistent features of GSAT and CSAT's average behaviour.

6 Improving GenSAT

One feature of GSAT's (and CSAT's) behaviour which was not described in the last section is their tendency to repeat flips. If GSAT is on a plateau and is forced to make a sideways move, then making the reverse sideways move to loop back will remain a possible flip, except on the rare occasions when the score can actually be increased after a sideways move. GSAT on fifty variable problems typically doubled back on itself on about 20% of its flips. Although there are rare occasions when such loops are essential, such flips are usually wasteful. To see if improved performance would result when these loops were eliminated, we implemented a variant of *pick* which will not return the last variable flipped unless there is no other choice. The resulting variant of GSAT we call "MSAT", where M stands for "memory". Some results for MSAT are summarized in the following table.

Problem	Procedure	Tries		Flips	
Random 50 Vars	MSAT	5.15	(85%)	92.3	(97%)
Random 100 Vars	MSAT	19.8	(79%)	264	(101%)
6-queens	MSAT	1.11	(48%)	55.1	(83%)
8-queens	MSAT	1.07	(92%)	50.5	(59%)
16-queens	MSAT	1.01	(99%)	159	(63%)

It can be seen that significant improvements were obtained over GSAT. Similar performance improvements were obtained adding loop checking to CSAT. This may also help to explain in part why DSAT performs better than GSAT since the fair deterministic *pick* used by DSAT automatically performs loop-checking.

The results for MSAT are particularly interesting in the light of the graphs presented in section 5. The average value of poss-flips is about 0.1N after about N flips, that is 5 variables if N=50. Since GSAT is typically offered 5 variables to flip, naturally it repeats itself about 20% of the time; this is the figure we obtained experimentally. Furthermore, we ran GSAT on 50 variable random problems with 40% more flips (repeated flips occur 20% of the time but each wastes 2 flips), and the results were indistinguishable from MSAT.

MSAT has a memory of just one flip; it therefore tries to avoid returning to positions it was at one flip ago. Any greater length of memory does not seem to help. With N=50, we are typically offered one of 5 variables to flip; thus, there is approximately a $\frac{1}{5}$ chance of a loop of length 2 but only a $\frac{1}{16}$ chance of a loop of length 4 (given that we prohibit loops of length 2). The probabilities for longer loops are even smaller. We implemented versions of MSAT with longer memories up to and including ESAT which has a memory of Max-flips.[5] Although we sometimes observed improved performance over MSAT, further experimentation is needed since no consistent picture has yet emerged.

To conclude, adding even limited memory to GSAT and CSAT can improve their performance significantly.

7 Related Work

Experimental studies of GSAT were first presented by [Selman *et al* 92]. A closely related set of procedures has also been studied by Gu [Gu 92, Gu & Gu 91]. For example, in SAT1.3 [Gu & Gu 91], as in fair but cautious DSAT, the variables are stepped through in order, each one being flipped if it increases the score. However, SAT1.3 has a different control structure to GenSAT which allows it to make sideways moves when upwards moves are possible. Similar comments apply to Gu's SAT6.0 [Gu 92] which is close to the greedy version of DSAT. Gu reports impressive results using these procedures, but they are not directly comparable to the results reported here since they are not for the difficult regions of random 3-SAT problems. We expect Gu's procedures to behave in a very similar fashion to the variants of GenSAT analysed here. Furthermore, we expect that they can be varied as flexibly as GenSAT.

Procedures like GenSAT have been successfully applied to constraint satisfaction problems other than satisfiability. For example, [Minton *et al* 90] have proposed a greedy local search procedure which performed well scheduling observations on the Hubble Space Telescope, and other constraint problems like the million-queens, and 3-colourability. It is interesting to note that the performance of this procedure was significantly improved by heuristics for choosing a good initial assignment.

[5] "E" is for "elephant" as ESAT never forgets.

GenSAT is an *iterative improvement* algorithm [vanLaarhoven & Aarts 87] which uses a distribution of initial assignments to overcome poor start positions and local maxima. *Simulated annealing* is an alternative approach to these problems but is probably of little use in GenSAT given the low density of local maxima, and the size of the plateaus.

8 Further Work

Although we have determined that the initial truth assignment needs to vary from try to try, we have not yet investigated how this variation is best achieved. For example, the initial assignments could be enumerated so that successive assignments are in completely different parts of the search space. Alternatively information about an unsuccessful try might guide the initial assignment for the next try. Another interesting possibility would be to construct the initial assignment in a greedy way, as in [Minton *et al* 90]. This might eliminate most of the climbing phase of GenSAT's search.

Two aspect of GenSAT that we have not varied in this paper are the scoring function and the control strategy. The score has always been the number of clauses satisfied. High percentage scores can be achieved very rapidly using many variants of GenSAT. However, this score function is insensitive to the nature of the search space once these high percentages are reached. We therefore intend to investigate alternative score functions that may be more effective once the plateau has been reached. Some studies in this direction have been undertaken in [Gu 92]. Alternative control structures to GenSAT are used in procedures proposed by [Gu 92]. We intend to investigate these procedures to see if their properties are similar to GenSAT's.

Finally to study all these issues further on larger problems, and to study the parallelizability of GenSAT, we intend to implement it on a highly parallel computer such as the Connection Machine.

9 Conclusions

In this paper, we have described a detailed study of some variants of GSAT, a greedy random hill-climbing procedure for satisfiability. To do this we introduced GenSAT, a generalisation of GSAT. GenSAT is able to solve large and difficult satisfiability problems beyond the range of conventional procedures like Davis-Putnam.

Our results show that there is nothing essential to GenSAT's performance in its greediness or randomness in choosing among equally scored choices: cautious and fair deterministic procedures work just as well. GenSAT's starting position must though be allowed to vary. Our results also show that hill-climbing is crucial to GenSAT's performance. However, climbing only takes place during the first short phase of the search; after this phase almost all flips make no improvement in the score. In addition, we have shown that performance can be considerably improved by the use of a limited amount of memory to avoid previously visited positions.

In investigations of GSAT's hill-climbing we presented graphs which showed remarkable consistency despite large variations in problem size. We cannot explain the exact positions of certain critical points on these graphs, such as the transition from climbing to plateau-searching. Nor can we yet explain why the graphs scale, on random problems at least, linearly with the problem size. Providing such explanations remains a fascinating research problem.

References

[Cook 71] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computation*, pages 151–158, 1971.

[Davis & Putnam 60] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. Association for Computing Machinery*, 7:201–215, 1960.

[Gu & Gu 91] Jun Gu and Qian Ping Gu. Average time complexities of several local search algorithms for the satisfiability problem (sat). Technical Report UCECE-TR-91-004, University of Calgary, Canada, 1991.

[Gu 92] Jun Gu. Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin*, 3(1), January 1992.

[Kautz & Selman 92] H.A. Kautz and B. Selman. Planning as Satisfiability. In *Proceedings of the 10th ECAI*, pages 359–363, 1992.

[Minton *et al* 90] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *AAAI-90, Proceedings Eighth National Conference on Artificial Intelligence*, pages 17–24. AAAI Press/MIT Press, 1990.

[Mitchell *et al* 92] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *AAAI-92: Proceedings Tenth National Conference on Artificial Intelligence*. AAAI Press/The MIT Press, July 12-16 1992.

[Reiter & Mackworth 89] R. Reiter and A. Mackworth. A logical framework for depiction and image interpretation. *Artificial Intelligence*, 41(3):123–155, 1989.

[Selman *et al* 92] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *AAAI-92: Proceedings Tenth National Conference on Artificial Intelligence*. AAAI Press/The MIT Press, July 12-16 1992.

[vanLaarhoven & Aarts 87] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, Dordrecht, Holland, 1987.