Towards an Understanding of Hill-climbing Procedures for SAT *

Ian P. Gent
Toby Walsh †

Draft of January 12, 1993

Abstract

Recently several local hill-climbing procedures for propositional satisfiability have been proposed, which are able to solve large and difficult problems beyond the reach of conventional algorithms like Davis-Putnam. By the introduction of some new variants of these procedures, we provide strong experimental evidence to support the conjecture that neither greediness nor randomness is important in these procedures. One of the variants introduced seems to offer significant improvements over earlier procedures. In addition, we investigate experimentally how their performance depends on their parameters. Our results suggest that run-time scales less than simply exponentially in the problem size.

1 Introduction

Recently several local hill-climbing procedures for propositional satisfiability have been proposed [4, 3, 11]. Propositional satisfiability (or SAT) is the problem of deciding if there is an assignment for the variables in a propositional formula that makes the formula true. SAT was one of the first problems shown to be NP-hard [1]. SAT is also of considerable practical interest as many AI tasks can be encoded quite naturally as SAT problems (*eg.* planning [5], constraint satisfaction, vision interpretation [9], refutational theorem proving). Much of the interest in these

*This research was supported by SERC Postdoctoral Fellowships to the authors.
†Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN; Email:I.P.Gent@edinburgh.ac.uk, T.Walsh@edinburgh.ac.uk. We thank Alan Bundy, Bob Constable, Judith Underwood and the members of the Mathematical Reasoning Group for their constructive comments and their estimated 100 trillion CPU cycles.

local hill-climbing procedures is because they scale well and because they can solve large and difficult SAT problems beyond the reach of conventional algorithms like the Davis-Putnam procedure [2].

These hill-climbing procedures share three common features. First, they attempt to determine the satisfiability of a formula in conjuctive normal form (CNF)[1]. Second, they hill-climb on the number of satisfied clauses. And third, their local neighbourhood (which they search for a better truth assignment) is the set of truth assigments with the assignment to *one* variable changed. Typical of such procedures is GSAT [11], a greedy random hill-climbing procedure. GSAT starts with a randomly generated truth assignment, and hill-climbs by changing (or "flipping") the variable assignment which gives the largest increase in the number of clauses satisfied. Given the choice between several equally good flips, it picks one at random.

In [3] we investigated some features of GSAT. In particular, we focused on three questions. Is greediness important? Is randomness important? Is hill-climbing important? One of the aims of this paper is to provide stronger and more complete answers to each of these three question. In particular, we will show that neither greediness nor randomness is important.

We will propose some new procedures which show considerably improved performance over GSAT on certain classes of problems. Finally, we will explore in more detail how these procedures scale, and how best to set their parameters. Since there is nothing particularly special about GSAT or the other procedures we analyse, we expect that our results will translate to any procedure which performs local hill-climbing on the number of satisfied clauses (for example SAT 1.1 and SAT 6.0 [4]). To perform these investigations, we use a generalisation of GSAT called "GenSAT" first introduced in [3].

procedure GenSAT(Σ)
    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 := *hill-climb(Σ, T)* ; compute best local neighbours
                    V := *pick*(Poss-flips) ; pick one to flip
                    T := T with V's truth assignment flipped
        end
    end
    return "no satisfying assignment found"

[1] A formula, Σ is in CNF iff it is a conjunction of clauses, where a clause is a disjunction of literals.

GSAT is a particular instance of GenSAT in which *initial* generates a random truth assignment, *hill-climb* 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* choses one of these variables at random. An important feature of GSAT's hill-climbing is sideways flips – if there exists no flip which increases the score, then a variable is flipped which does not change the score. GSAT's performance degrades greatly without sideways flips.

2 Greediness and Hill-climbing

To study the importance of greediness, we introduced CSAT [3], a cautious variant of GenSAT. In CSAT, *hill-climb* returns all variables which increase the score when flipped, or if there are no such variables, all variables which make no change to the score, or if there are none of these, all variables. Since we found no problem sets on which CSAT performed significantly worse than GSAT, we conjectured that greediness is not important [3]. To test this conjecture, we introduce three new variants of GenSAT: TSAT, ISAT, and SSAT.

TSAT is timid since *hill-climb* returns those variables which increase the score the least when flipped, or if there are no variables which increase the score, all variables which make no change, or if there are none of these, all variables. ISAT is indifferent to upwards and sidways flips since *hill-climb* returns those variables which do not decrease the score when flipped, or if there are none of these, all variables. SSAT, on the other hand, is a sideways moving procedure since *hill-climb* returns those variables which make no change to the score when flipped, or if there are no such variables, all those variables which increase the score, or if there are none of these, all variables. The results for these procedures are given in table 1.

As in [3], we test these procedures on two types of problems: satisifability encodings of the n-queens problems, and random k-SAT. The n-queens problem is to place n-queens on an $n \times n$ chessboard so that no two queens attack each other. Its encoding as a SAT problem use n^2 variables, each true iff a particular square is occupied by a queen. Problems in random k-SAT with N variables and L clauses are generated 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 $\frac{1}{2}$. For random 3-SAT the ratio L/N = 4.3 has been identified as giving problems which are particularly hard for Davis-Putnam and many other algorithms [8, 6]. This ratio was also used in an earlier study of GSAT [11] and in [3]. Note that because GenSAT variants typically do not determine unsatisfiablity, unsatisfiabile formulas were filtered out by the Davis-Putnam procedure.

In every experiment in this paper (unless explicitly mentioned otherwise) Max-flips was set at 5 times the number of variables while Max-tries was set large enough

Problem	Procedure	Tries	Flips	Total Flips	s.d.
Random	GSAT	5.87	93.8	1310	2200
50 vars	TSAT	5.32	96.4	1180	2090
	ISAT	6.35	127	1460	2560
Random	GSAT	10.7	158	3550	6090
70 vars	TSAT	10.2	161	3390	5980
	ISAT	11.9	208	4030	7890
Random	GSAT	25.7	261	12600	22800
100 vars	TSAT	26.1	272	12800	22000
	ISAT	34.6	327	17100	43200
6-queens	GSAT	2.14	65.0	271	267
	TSAT	2.26	74.1	301	296
	ISAT	2.22	78.8	298	310
8-queens	GSAT	1.18	84.5	141	170
	TSAT	1.20	101	165	171
	ISAT	1.21	112	178	173
16-queens	GSAT	1.03	253	288	251
	TSAT	1.04	282	326	295
	ISAT	1.02	339	365	226

Table 1: Comparison of GSAT, TSAT, and ISAT

to allow all experiments to succeed. The figures for tries are the average number of tries taken until success (and is therefore at least 1), while the figures for flips give the average number of flips in successful tries only. 1000 experiments were performed in each case, all of which were successful. For each random problem class, we performed all experiments in this paper on the same set of randomly generated problems; this reduces the variance between the results.

The results in table 1 confirm our conjecture that greediness is not important. Like cautious hill-climbing [3], timid hill-climbing gives very similar performance to greedy hill-climbing. The differences between GSAT and TSAT in table 1 are less than variances we have observed on problem sets of this size. ISAT does, however, perform significantly worse than GSAT. ISAT's performance falls off much more quickly as the problem size increases. We conjecture that as the problem size increases, the number of sideways flips offered increases and these are typically poor moves compared to upwards flips. Combined with other heuristics, however, some of these sideways flips can be good flips to make. In section 4, we will introduce a variant of ISAT which can give improved performance over GSAT. As well as SSAT, we tried a variant of ISAT which is indifferent to flips which increase the score, leave it constant and decrease it by 1. Both this variant and SSAT failed to solve any of 25 random 3-SAT 50 variable problems in 999 tries.

We therefore conclude that you need to perform some sort of hill-climbing.

Greediness has also proved useful in several local search procedures for the generation of start positions (*eg.* in a constraint satisfaction procedure [7], and in various algorithms for the n-queens problems [12]). To investigate whether such initial greediness would be useful for satisfiability, we introduce a new variant of GenSAT called OSAT which is opportunistic in its generation of an initial truth assignment. In OSAT, the score function (number of satisfied clauses) is extended to partial truth assignments by ignoring unassigned variables. OSAT incrementally builds an initial truth assignment by considering the variables in some random order and picking those truth values which maximize the score; considering the variables in a random order helps prevent any variable from dominating. In addition, if the score is identical for the assignment of a variable to true and false, a truth assignment is chosen at random. OSAT is identical to GSAT in all other respects. A comparison of OSAT and GSAT is given in table 2. (Figures in brackets give the Total Flips figure as a percentage of the comparable figure for GSAT.)

Problem	Procedure	Tries	Flips	Total Flips	(% GSAT)	s.d.
Random 50 vars	OSAT	6.82	78.6	1530	(120%)	4750
Random 70 vars	OSAT	9.50	139	3110	(88%)	7570
Random 100 vars	OSAT	32.6	235	16000	(130%)	74800
6-queens	OSAT	2.15	62.0	270	(100%)	298
8-queens	OSAT	1.18	67.8	126	(89%)	161
16-queens	OSAT	1.02	145	165	(57%)	211

Table 2: Comparison of GSAT and OSAT

OSAT always takes less flips on average than GSAT on a successful try. OSAT also takes the same or slightly more tries as GSAT. The total number of flips performed by OSAT can therefore be slightly less than GSAT on the same problems. However, if we include the O(N) computation necessary to perform the greedy start, OSAT is nearly always slower than GSAT.

To conclude, our results confirm that greediness is neither important in hill-climbing nor in the generation of the initial start position. Any form of hill-climbing which prefers up or sideways moves over downwards moves (and does not prefer sideways over up moves) appears to work.

3 Randomness

GSAT uses randomness in generating the initial truth assignment and in picking which variable to flip when offered more than one. To explore the importance of such randomness, we introduced in [3] three variants of GenSAT: FSAT, DSAT, and USAT. FSAT uses a fixed initial truth assignment but is otherwise identical to GSAT. DSAT picks between equally good variables to flip in a deterministic but fair way, whilst USAT picks between equally good variables to flip in a deterministic but unfair way[2]. On random k-SAT problems both USAT and FSAT performed poorly. DSAT, however, performed considerably better than GSAT. We therefore concluded that there is nothing essential about the randomness of picking in GSAT (although fairness is important) and that the initial truth assignment must vary from try to try.

To explore whether the initial truth assignment can be varied deterministically, and to determine if randomness can be eliminated simultaneously from all parts of GenSAT, we introduce three new variants: NSAT, VSAT, and VDSAT. NSAT generates initial truth assignments in "numerical" order. That is, on the n-th try, the m-th variable in a truth assignment is set to true iff the m-th bit of the binary representation of n is 1. VSAT, by comparison, generates initial truth assignments to maximize the variability between successive assignments. On the first try, all variables are set to false. On the second try, all variables are set to true. On the third try, half the variables are set to true and half to false, and so on. The exact algorithm used to generate assignments is given in the Appendix. Since this algorithm cycles through all possible truth assignments, VSAT is a complete decision procedure for SAT when Max-tries is set to 2^N. NSAT and VSAT are identical to GSAT in all other respects. VDSAT uses the same start function as VSAT and is identical to DSAT in all other respects. Unlike all previous variants, VDSAT is entirely deterministic.

As table 3 demonstrates, NSAT's performance was very poor on 50 variable problems. We conjecture thst this poor performance is a consequence of the lack of variability between successive initial truth assignments. VSAT and VDSAT have initial truth assignments which vary much more than initial truth assignments in NSAT. VSAT's performance is very close to GSAT's. VDSAT performs very similarly to DSAT, and better than GSAT. Note that the results for VDSAT on queens problems are not averages but exact since VDSAT's performance is entirely determined once the problem is specified.

To conclude, randomness is neither important in the initial start position nor in the picking between equally good variables. It is important, however, that successive initial truth assignments vary on a large number of variables. In section 4 we will introduce a new and deterministic variant of GenSAT which supports this hypothesis.

[2] We call a variant of GenSAT "fair" if it will eventually pick any variable that is offered continually. USAT always picks the least variable in a fixed ordering. DSAT picks variables in an order which cycles.

Problem	Procedure	Tries	Flips	Total flips	(% GSAT)	s.d.
Random	VSAT	6.18	91.6	1390	(110%)	2370
50 vars	DSAT	4.79	71.5	1020	(78%)	2040
	VDSAT	4.32	74.1	904	(69%)	2070
	NSAT	40.1	106	9870	(750%)	46400
Random	VSAT	10.4	155	3440	(97%)	5710
70 vars	DSAT	6.82	123	2160	(61%)	3410
	VDSAT	6.90	124	2190	(62%)	3950
Random	VSAT	30.4	270	14900	(120%)	36400
100 vars	DSAT	15.2	227	7350	(58%)	16500
	VDSAT	14.7	227	7090	(56%)	16300
6-queens	VSAT	2.27	76.0	305	(110%)	346
	DSAT	1.09	46.1	61.6	(23%)	60.2
	VDSAT	2	50	230	(85%)	—
	NSAT	2.07	65.1	258	(95%)	275
8-queens	VSAT	1.17	77.5	132	(94%)	160
	DSAT	1.11	45.8	80.7	(57%)	110
	VDSAT	1	30	30	(21%)	—
	NSAT	1.17	74.4	128	(91%)	147
16-queens	VSAT	1.03	160	196	(68%)	299
	DSAT	1.03	155	198	(69%)	242
	VDSAT	2	296	1576	(550%)	—
	NSAT	1.03	156	190	(66%)	252

Table 3: Comparison of VSAT, DSAT, VDSAT, NSAT and GSAT

4 Memory

Information gathered during a run of GenSAT can be used to guide future search. For example, Selman and Kautz [10] have introduced a variant of GSAT in which a failed try is used to weight the emphasis given to clauses by the score function in future tries. They report that this technique enables GSAT to solve problems that it otherwise cannot solve.

In [3] we introduced MSAT, which is like GSAT except that it uses memory to avoid making the same flip twice in a row except where climbing gives no other choice. MSAT showed improved performance over GSAT particularly on the n-queens problem, although the improvement declines as problems grow larger in size. This is, of course, not the only way we can use memory of the earlier search. In this section we introduce HSAT, and IHSAT. These variants of GenSAT use historical information to choose deterministically which variable to pick. When offered a choice of variables, HSAT always picks the one that was flipped longest ago (in the current try): if two variables are offered which have never been flipped in this try, an arbitrary (but fixed) ordering is used to choose between them. HSAT is otherwise like GSAT. IHSAT uses the same *pick* as HSAT but is indifferent like ISAT. Results for HSAT and IHSAT are summarised in table 4.

Problem	Procedure	Tries	Flips	Total Flips	(% GSAT)	s.d.
Random	HSAT	3.82	58.7	763	(58%)	1660
50 vars	IHSAT	3.38	96.0	690	(53%)	1490
Random	HSAT	4.93	101	1480	(42%)	2510
70 vars	IHSAT	3.84	165	1160	(33%)	1900
Random	HSAT	8.11	184	3740	(30%)	7770
100 vars	IHSAT	6.95	274	3250	(26%)	6480
6-queens	HSAT	1.11	43.3	62.9	(23%)	68.7
	IHSAT	1.08	55.8	70.6	(26%)	58.1
8-queens	HSAT	1.09	44.1	73.9	(52%)	110
	IHSAT	1.08	66.2	90.5	(64%)	95.2
16-queens	HSAT	1.02	156	183	(64%)	190
	IHSAT	1.02	220	245	(85%)	183

Table 4: Comparison of HSAT and IHSAT

Both HSAT and IHSAT perform considerably better than GSAT. Indeed, both perform better than any previous variant of GenSAT. Many other variants of HSAT also perform very well (*eg.* HSAT with cautious hill-climbing, with timid hill-climbing, with VSAT's start function). Note also that, unlike MSAT, the improvement in performance does not appear to decline as the number of variables increases.

HSAT picks based on when variables were last flipped: a natural variant uses instead the last time that variables were returned by *hill-climb* whether or not they were picked. Since *hill-climb* can return a set of variables, a method is needed to pick one variable if a number were offered equally long ago. We implemented three such versions of HSAT using random, deterministic and historical methods for this subsidiary picking. The performance of the random and deterministic methods was not as good as HSAT, whilst the the last method seemed to offer closely comparable performance to HSAT. To date we have not observed any of these variants offering a significant performance improvement over HSAT.

To conclude, memory of the current try can significantly improve the performance of many variants of GenSAT. In particular, picking variables based on the history of the try rather than randomly is one such improvement.

5 Running GenSAT

We have studied the behaviour of GenSAT as the functions *initial*, *hill-climb*, and *pick* are varied. However, we have not discussed the behaviour of GenSAT as we vary its explicit parameters, Max-tries and Max-flips. The setting of Max-tries is quite simple – it depends only on one's patience. Increasing Max-tries will increase one's chance of success. Indeed, since all our experiments are on satisfiable problem sets, we have often set Max-tries to infinity.

The situation for Max-flips is rather different to that for Max-tries. Although increasing Max-flips increases the probability of success on a given try, it can decrease the probability of success in a given run time. To understand this fully it is helpful to review some features of GenSAT's search space identified in [3]. GenSAT's hill-climbing is initially dominated by increasing the number of satisfied clauses. GSAT, for example, on random 3-SAT problems is typically able to climb for about 0.25N flips, where N is the number of variables in the problem, increasing the percentage of satisfied clauses from 87.5% ($\frac{7}{8}$ of the clauses are initially satisfied by a random assignment) to about 97%. From this point on, there is little climbing; the vast majority of flips are sideways, neither increasing nor decreasing the score. Occasionally a flip can increase the score. On some tries, this happens often enough before Max-flips is reached that all the clauses are satisfied.

In Figure 1, we have plotted the percentage of problems solved against the total numbers of flips used by HSAT for 50 variable random problems, with Max-flips = 150. The dotted lines represent the points when new tries were started. During the initial climbing phase almost no problems are solved: in fact no problems were solved in less that 10 flips on the first try. Note that 10 is 0.2N, approximately the length of the initial climbing phase. This behaviour is repeated during each try: very few problems are solved during the first 10 flips of a try. After about 10 flips, there is a dramatic change in the gradient of the graph. There is now a significant chance of solving a problem with each flip. Again, this behaviour is repeated on each try. Finally, after about 100 flips of a given try, the gradient declines noticeably. From now on, there is a very small chance of solving a problem during the current try if it has not been solved already.

In Figure 2, we have added the comparable graph with Max-flips = 75. The performance over the first 75 flips is identical. After this, the first experiment with Max-flips = 150 continues to solve problems on the first try, but the second experiment with Max-flips = 75 starts another try, and hence fails to solve any problems for a short period. However, it very soon enters a highly productive phase and overtakes the experiment with Max-flips = 150. These graphs suggest that 75 is a better setting for Max-flips for HSAT on these problems than 150.

To determine the optimal values for Max-flips, we have plotted in Figure 3 the average total number of flips used on 50 variable problems against integer values of Max-flips from 25 to 300 for three variants of GenSAT: HSAT, DSAT, and

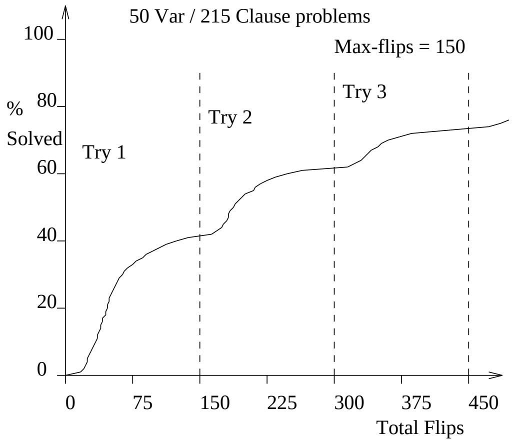

Figure 1: HSAT, Max-flips = 150

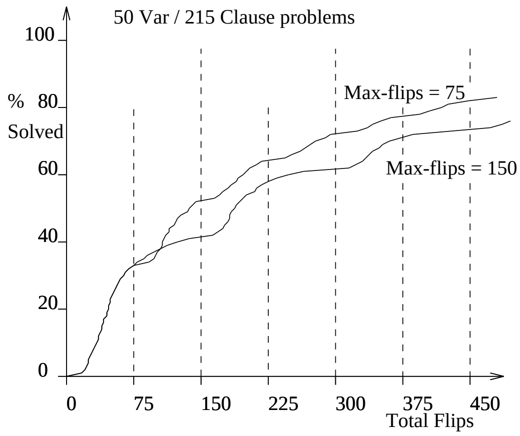

Figure 2: HSAT, Max-flips = 75, 150

GSAT. The total number of flips used is a measure of computational resources. Note that it does not bear a simple linear relationship with cpu time as there is a start-up cost of O(N) associated with each try. For small values of Max-flips, not enough flips remain after the hill-climbing phase to give a high chance of success on each try. Each variant performs much the same which is to be expected as each is performing the same (greedy) hill-climbing. The optimum value for Max-flips is about 60. Since this minimum is not very sharp, it is not, however, too important to find the exact optimal value. For Max-flips larger than about 100, the later flips of most tries are unsuccessful and hence lead to wasted work. As Max-flips increases, the amount of wasted work increases almost linearly. For everything but small values of Max-flips, HSAT takes fewer flips than DSAT, which in turn takes fewer than GSAT. The type of picking performed thus seems to have a significant effect on the chance of success in a try if more than a few flips are needed.

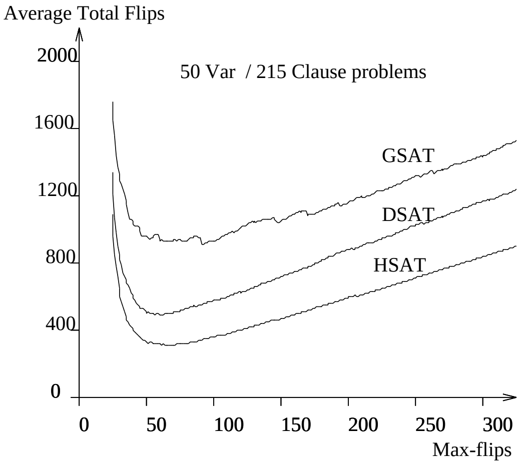

Figure 3: Varying Max-flips

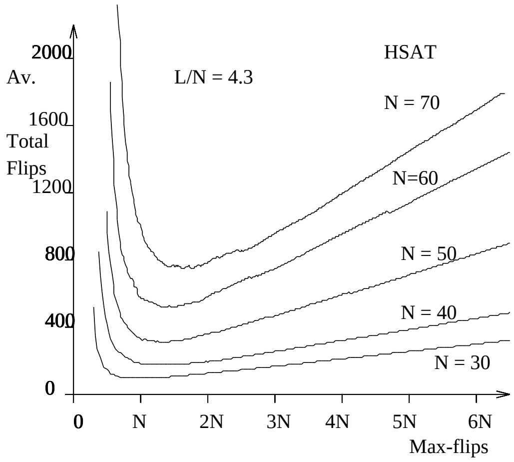

Figure 4: Varying Max-flips & N

Similar results are observed when the problem size is varied. In Figure 4, we have plotted the average total number of flips used by HSAT on random problems against integer values of Max-flips for differing numbers of variables N. The optimal value of Max-flips appears to increase approximately as N^2. Even with 100 variable random problems, the optimal value is, however, only about 2N flips. Figure 4 also supports the claim made in [11] and [3] that these hill-climbing procedures appear to scale better than conventional procedures like Davis-Putnam.

To investigate more precisely how various GenSAT variants scale, Figure 5 gives the average total number of flips used by GSAT, DSAT and HSAT on random problems against the number of variables N (at 10 variable intervals from 10 to 100). Although the average total flips increases rapidly with N, the rate of growth seems to be less than a simple exponential. In addition, the improvement in performance offered by HSAT over DSAT, and by DSAT over GSAT increases greatly with N. One cause of variability in these results is that Max-flips is set to 5N and not its optimal value. In figure 6, we have therefore plotted the *optimal* values for the average total flips against the number of variables again at 10 variable intervals. For clarity, the average total flips is plotted on a log scale. The performances of GSAT, DSAT and HSAT in Figure F are consistent with a small (less than linear) exponential dependence on N. Note that the data does not rule out a polynomial dependency on N of about order 3. Further experimentation and a more complete theoretical understanding are needed to choose between these two interpretations. We can, however, observe (as have Selman, Kautz and Mitchell [11]) that these hill-climbing procedures have solved some large and difficult random 3-SAT problems well beyond the reach of conventional procedures. At worse, their behaviour appears to be exponential with a small exponent. Note also that the improvement in performance offered by HSAT over DSAT, and by DSAT over GSAT increases with N. Procedures like HSAT therefore offer real advantages over GSAT and DSAT, not just constant factor speed-ups.

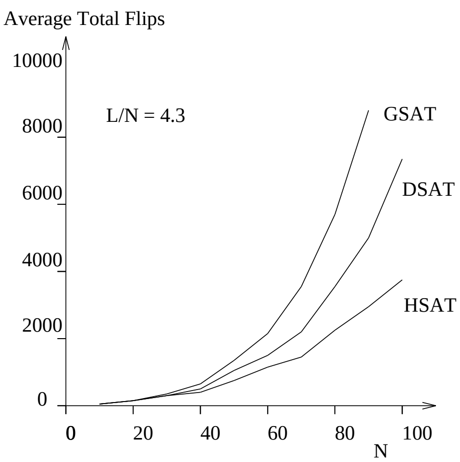

Figure 5: Max-flips = 5N

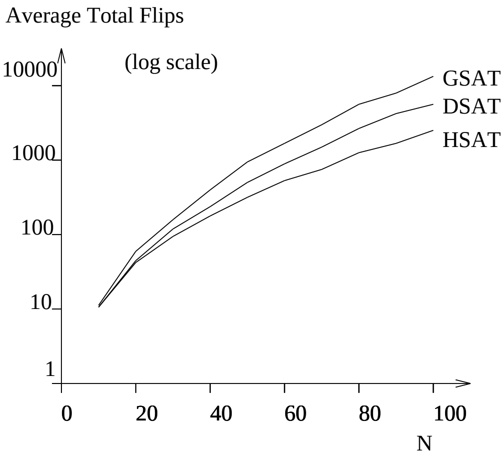

Figure 6: Max-flips Optimal

6 Related and Future Work

Hill-climbing search has been used in many different domains, both practical (*eg.* scheduling) and artificial (*eg.* toy problems like the 8-puzzle). Only recently, however, has hill-climbing been applied to SAT. Some of the first procedures to hill-climb on the number of satisfied clauses were proposed in [4]. GSAT also hill-climbs on the number of satisfied clauses and was first presented in [11]. Unfortunately, it is difficult to compare these procedures directly as they use different control structures. One common problem with hill-climbing is escaping local maxima. Although simulated annealing has often proved successful at tackling this problem, it is probably of little use in GenSAT given the low density of local maxima, and the use of different start positions.

These experiments have been performed with just two types of SAT problems: random k-SAT for $k = 3$ and $L/N = 4.3$, and an encoding of the n-queens. Although we expect that similar results would be obtained with other random and structured problem sets, we intend to confirm this conjecture experimentally. In particular, we would like to try other values of k and L/N, and other non-random problems (*eg.* blocks world planning encoded as SAT [5], boolean induction, standard graph colouring problems encoded as SAT). To test problem sets with large numbers of variables, we intend to implement GenSAT on a Connection Machine. This will be an interesting exercise as GenSAT appears to have a large degree of parallelizability.

Two aspects of GenSAT that we have not probed in detail are the control structure and the scoring function. Alternative control structures for hill-climbing on the number of satsified clauses are proposed in [4]. We intend to perform some experiments to determine if such control structures give rise, as we expect, to similar performance. In GenSAT the score function has always been the number of clauses satisfied. Since much of the search consists of sideways flips, this score function is perhaps a little insensitive. We therefore intend to investigate alternative score functions. Finally, we would like to develop a better theoretical understanding of these experimental results. Unfortunately, as with simulated annealing, we fear that such a theoretical analysis may be rather difficult to construct.

7 Conclusions

Recently, several local hill-climbing procedures for propositional satisfiability have been proposed [11, 3]. In [3], we conjectured that neither greediness nor randomness was essential for the effectiveness of the hill-climbing in these procedures. By the introduction of some new variants, we have confirmed this conjecture. Any (random or fair deterministic) hill-climbing procedure which prefers up or sideways moves over downwards moves (and does not prefer sideways over up moves) appears to work. In addition, we have shown that randomness is not essential for generating the initial start position, and that greediness here is actually counter-productive. We have also proposed a new variant, HSAT, which performs much better than previous procedures on our problem sets. Finally, we have studied in detail how the performance of these procedures depends on the setting of their parameters. At worst, our experimental evidence suggests that they scale with a small (less than linear) exponential dependence on the problem size. This supports the conjecture made in [11] that such procedures scale well and can be used to solve large and and difficult SAT problems beyond the reach of conventional algorithms.

Appendix

The initial truth assignment generated by VSAT uses a successive binary division on the variables. VSAT therefore ideally needs 2^M variables. Given a number of variables N which is not a power of 2, VSAT generates a truth assignment for 2^M variables where 2^M is the smallest integer power of 2 equal to or bigger than N and truncates to the first N assignments. Let $vstart(M, p)$ be the truth assigment given to 2^M variables at the p-th try by VSAT. Truth assignments will be represented by lists of truth values.

The function *vstart* uses a simple recursion on M. For the base case ($M = 0$, *ie.* 1 variable), $vstart(0, p)$ is $[false]$ if p is even, and $[true]$ otherwise. This is, of course, maximally variable. For the step case, we divide the 2^M variables into two sets of 2^{M-1} variables. We assume that we can assign truth values in some maximally variable way to 2^{M-1} variables (with a cycle of length $2^{2^{M-1}}$). We assign both sets of variables using this cycle. With the first set, however, we rotate through the cycle every time we go through it completely (that is, after every $2^{2^{M-1}}$ calls to vstart). Thus,

$$vstart(M, p) \quad = \quad vstart(M-1, p+r) <> vstart(M-1, p)$$

where $r = floor(\frac{p}{2^{2^{M-1}}})$ and $<>$ is the infix list append operator. This function cycles through all possible truth assignments.

References

[1] 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.

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

[3] I. Gent and T. Walsh. The Enigma of SAT Hill-climbing Procedures. Technical Report 605, Dept. of Artificial Intelligence, University of Edinburgh, 1992. Under review for IJCAI-93.

[4] J. Gu. Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin*, 3(1):8–12, 1992.

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

[6] T. Larrabee and Y. Tsuji. Evidence for a Satisfiability Threshold for Random 3CNF Formulas. Technical Report UCSC-CRL-92-42, Baskin Center for Computer Enigineering and Information Sciences, University of California, Santa Cruz, 1992.

[7] 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.

[8] 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.

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

[10] B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. Technical report, Artificial Intelligence Principles Research, AT & T Bell Laboratories, Murray Hill, NJ, 1993.

[11] B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the 10th National Conference on AI*, pages 440–446. American Association for Artificial Intelligence, 1992.

[12] Rok Sosič and Jun Gu. Fast search algorithms for the N-queens problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1572–1576, November/December 1991.