Towards an Understanding of Hill-climbing Procedures for SAT *

Ian P. Gent and **Toby Walsh**
Department of Artificial Intelligence, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, United Kingdom
Email:I.P.Gent@edinburgh.ac.uk, T.Walsh@edinburgh.ac.uk

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 our 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 performance depends on their parameters. Our results suggest that runtime scales less than simply exponentially in the problem size.

Introduction

Recently several local hill-climbing procedures for propositional satisfiability have been proposed [Gent and Walsh, 1992; Gu, 1992; Selman *et al.*, 1992]. 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 [Cook, 1971]. SAT is of considerable practical interest as many AI tasks can be encoded quite naturally into it (*eg.* planning [Kautz and Selman, 1992], constraint satisfaction, vision interpretation [Reiter and Mackworth, 1989], refutational theorem proving). Much of the interest in these procedures is because they scale well and can solve large and difficult SAT problems beyond the reach of conventional algorithms like Davis-Putnam.

These hill-climbing procedures share three common features. First, they attempt to determine the satisfiability of a formula in conjunctive normal form (a conjunction of clauses, where a clause is a disjunction of literals). Second, they hill-climb on the number of satisfied clauses. Third, their local neighbourhood (which they search for a better truth assignment) is the set of truth assignments with the assignment to *one* variable changed. Typical of such procedures is GSAT [Selman *et al.*, 1992], 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 equally good flips, it picks one at random.

In [Gent and Walsh, 1992] we investigated three features of GSAT. Is greediness important? Is randomness important? Is hill-climbing important? One of our aims is to provide stronger and more complete answers to these questions. We will show that neither greediness nor randomness is important. We will also propose some new procedures which show considerably improved performance over GSAT on certain classes of problems. Finally, we will explore how these procedures scale, and how to set their parameters. As there is nothing very 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 SAT1.1 and SAT6.0 [Gu, 1992]). To perform these experiments, we use a generalisation of GSAT called "GenSAT" [Gent and Walsh, 1992].

procedure GenSAT(Σ)
    for i := 1 **to** Max-tries
        T := *initial*(Σ) ; generate a 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 assignment flipped
        end
    end
return "no satisfying assignment found"

GSAT is an instance of GenSAT in which *initial* generates a random truth assignment, *hill-climb* returns those variables whose truth assignment if flipped gives

*This research was supported by SERC Postdoctoral Fellowships to the authors. 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.

the greatest increase in the number of clauses satisfied (called the "score" from now on) and *pick* chooses one of these variables at random. An important feature of GSAT's hill-climbing is sideways flips – if there is no flip which increases the score, a variable is flipped which does not change the score. GSAT's performance degrades greatly without sideways flips.

Greediness and Hill-climbing

To study the importance of greediness, we introduced CSAT [Gent and Walsh, 1992], 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 [Gent and Walsh, 1992]. 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 sideways 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, however, 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.

We test these procedures on two types of problems: satisfiability encodings of the n-queens 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 uses 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: a random subset of size k of the N variables is selected for each clause, and each variable 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 [Mitchell *et al.*, 1992; Larrabee and Tsuji, 1992]. This ratio was also used in an earlier study of GSAT [Selman *et al.*, 1992]. Since GenSAT variants typically do not determine unsatisfiability, unsatisfiable formulas were filtered out by the Davis-Putnam procedure.

In every experiment (unless explicitly mentioned otherwise) Max-flips was set to 5 times the number of variables and Max-tries to infinity. In Table 1, the figures for "Tries" are the average number of tries taken until success, while the figures for "Flips" give the average number of flips in successful tries only. The final two columns record the total number of flips (including

Problem	Proc	Tries	Flips	Total	s.d.
Random 50 vars	GSAT	5.87	93.8	1310	2200
	TSAT	5.32	96.4	1180	2090
	ISAT	6.35	127	1460	2560
Random 70 vars	GSAT	10.7	158	3550	6090
	TSAT	10.2	161	3390	5980
	ISAT	11.9	208	4030	7890
Random 100 vars	GSAT	25.7	261	12600	22800
	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

unsuccessful tries) and their standard deviations. 1000 experiments were performed in each case, all of which were successful. To reduce variance, all experiments used the same randomly generated problems.

The results in table 1 confirm our conjecture that greediness is not important. Like cautious hill-climbing [Gent and Walsh, 1992], timid hill-climbing gives very similar performance to greedy hill-climbing. The differences between GSAT and TSAT 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, as we show in a later section where a variant of ISAT gives 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 or 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 been used in several local search procedures for the generation of start positions (*eg.* in a constraint satisfaction procedure [Minton *et al.*, 1990], and in various algorithms for the n-queens problems [Sosič and Gu, 1991]). 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 a random order and picking those truth values which maximize the score; the use of 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. In this and subsequent tables, percentages give the total flips as a percentage of the comparable figure for GSAT, and standard deviations are omitted for reasons of space.

Problem	Proc	Tries	Flips	Total	%
Rand 50 vars	OSAT	6.82	78.6	1530	120%
Rand 70 vars	OSAT	9.50	139	3110	88%
Rand 100 vars	OSAT	32.6	235	16000	130%
6 queens	OSAT	2.15	62.0	270	100%
8 queens	OSAT	1.18	67.8	126	89%
16 queens	OSAT	1.02	145	165	57%

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.

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 [Gent and Walsh, 1992] 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[1]. On random k-SAT problems both USAT and FSAT performed poorly. DSAT, however, performed considerably better than GSAT (as well as [Gent and Walsh, 1992] see figures 4 & 5). 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.

[1] A procedure is *fair* if it eventually picks any variable that is offered continually. USAT picks the least variable in a fixed ordering. DSAT picks variables in a cyclical order.

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, 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. See [Gent and Walsh, 1993] for details. 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. Its performance on larger problems was even worse. We conjecture that 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, on random problems. VDSAT's performance on the 16 queens problem is poor because VDSAT is entirely deterministic and the first try happens to fail.

Problem	Proc	Tries	Flips	Total	%
Random 50 vars	NSAT	40.1	106	9870	750%
	VSAT	6.18	91.6	1390	110%
	VDSAT	4.32	74.1	904	69%
Random 70 vars	VSAT	10.4	155	3440	97%
	VDSAT	6.90	124	2190	62%
Random 100 vars	VSAT	30.4	270	14900	120%
	VDSAT	14.7	227	7090	56%
6 queens	NSAT	2.07	65.1	258	95%
	VSAT	2.27	76.0	305	110%
	VDSAT	2	50	230	85%
8 queens	NSAT	1.17	74.4	128	91%
	VSAT	1.17	77.5	132	94%
	VDSAT	1	30	30	21%
16 queens	NSAT	1.03	156	190	66%
	VSAT	1.03	160	196	68%
	VDSAT	2	296	1576	550%

Table 3: Comparison of NSAT, VSAT, and VDSAT

To conclude, randomness is neither important in the initial start position nor in picking between equally good variables. However, it is important that successive initial start positions vary on a large number of variables.

Memory

Information gathered during a run of GenSAT can be used to guide future search. For example, [Selman and Kautz, 1993] 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 [Gent and Walsh, 1992] we introduced MSAT, which is like GSAT except that it uses memory to avoid making the same flip twice in a row except when given no other choice. MSAT showed improved performance over GSAT particularly on the n-queens problem, although the improvement declines as problems grow larger. 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	Proc	Tries	Flips	Total	%
Random	HSAT	3.82	58.7	763	58%
50 vars	IHSAT	3.38	96.0	690	53%
Random	HSAT	4.93	101	1480	42%
70 vars	IHSAT	3.84	165	1160	33%
Random	HSAT	8.11	184	3740	30%
100 vars	IHSAT	6.95	274	3250	26%
6 queens	HSAT	1.11	43.3	62.9	23%
	IHSAT	1.08	55.8	70.6	26%
8 queens	HSAT	1.09	44.1	73.9	52%
	IHSAT	1.08	66.2	90.5	64%
16 queens	HSAT	1.02	156	183	64%
	IHSAT	1.02	220	245	85%

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.

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

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.

The situation for Max-flips is 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 identified in [Gent and Walsh, 1992]. 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 now 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 is 150. The dotted lines represent the start of new tries. 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.

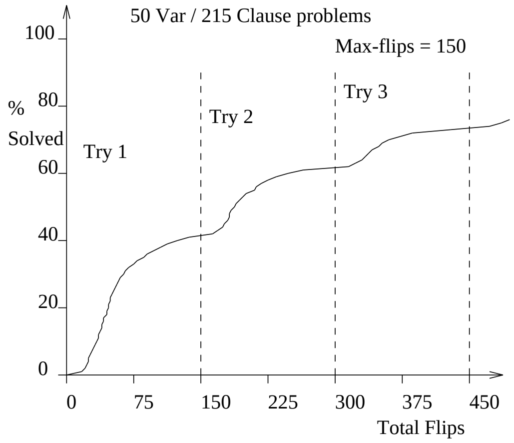

Figure 1: HSAT, Max-flips = 150

Different values of Max-flips offer a trade-off between the unproductive initial phase at the start of a try and the unproductive phase at the end for large Max-flips. To determine the optimal value, we have plotted in Figure 2 the average total number of flips used on 50 variable problems against integer values of Max-flips from 25 to 300 for HSAT, DSAT, and GSAT. 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. This 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.

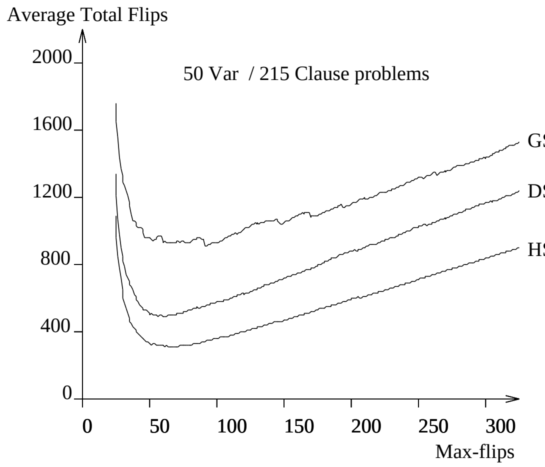

Figure 2: Varying Max-flips

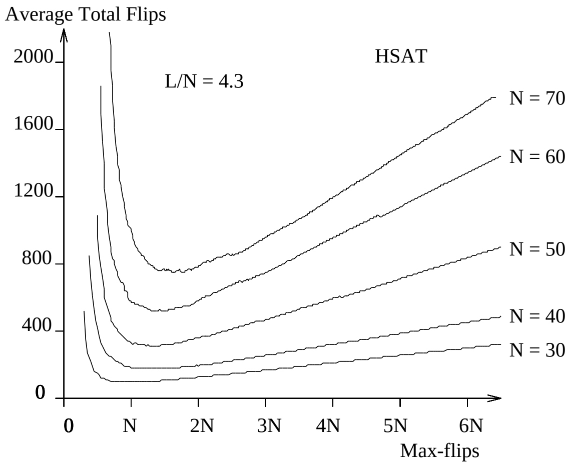

Figure 3: Varying Max-flips and N

Similar results are observed when the problem size is varied. In Figure 3, 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 only about 2N flips. Figure 3 also supports the claim made in [Selman *et al.*, 1992] and [Gent and Walsh, 1992] 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 4 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. 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 5 we have therefore plotted the *optimal* values of the average total flips against the number of variables at 10 variable intervals using a log scale for clarity. The performances of GSAT, DSAT and HSAT in Figure 5 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 do [Selman *et al.*, 1992]) that these hill-climbing procedures have solved some large and difficult random 3-SAT problems well beyond the reach of conventional procedures. At worst, their behaviour appears to be exponential with a small exponent. Note again that HSAT offers a real performance advantage over GSAT and DSAT, not just a constant factor speed-up.

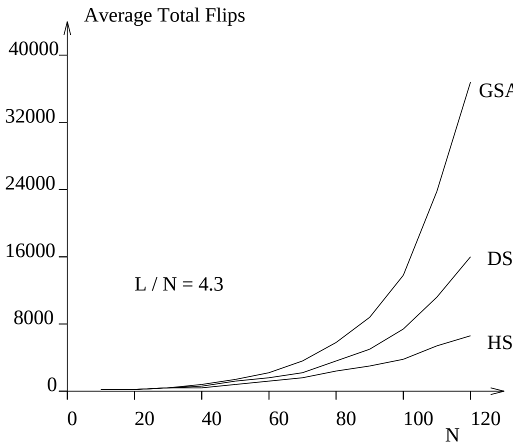

Figure 4: Max-flips = 5N

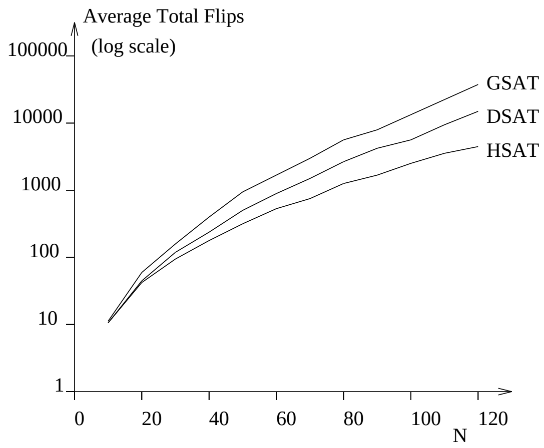

Figure 5: Max-flips Optimal

Related and Future Work

Hill-climbing search has been used in many different domains, both practical (*eg.* scheduling) and artificial (*eg.* 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 [Gu, 1992]. Unfortunately, it is difficult to compare these procedures with GenSAT directly as they use different control structures.

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 [Kautz and Selman, 1992], 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 be highly parallelizable.

One aspect of GenSAT that we have not probed in detail is the scoring function. The score function has always been the number of clauses satisfied. Since much of the search consists of sideways flips, this 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.

Conclusions

Recently, several local hill-climbing procedures for propositional satisfiability have been proposed [Selman *et al.*, 1992; Gent and Walsh, 1992]. In [Gent and Walsh, 1992], 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 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 [Selman *et al.*, 1992] that such procedures scale well and can be used to solve large and and difficult SAT problems beyond the reach of conventional algorithms.

References

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

Gent, I. and Walsh, T. 1992. The Enigma of SAT Hill-climbing Procedures. Research Paper 605, Dept. of AI, University of Edinburgh.

Gent, I. and Walsh, T. 1993. Towards an Understanding of Hill-climbing Procedures for SAT. Research Paper, Dept. of AI, University of Edinburgh.

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

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

Larrabee, T. and Tsuji, Y. 1992. Evidence for a Satisfiability Threshold for Random 3CNF Formulas. Technical Report UCSC-CRL-92-42, UC Santa Cruz.

Minton, S.; Johnston, M.; Philips, A.; and Laird, P. 1990. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proc. 8th National Conference on AI*. 17–24.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *Proc. 10th National Conference on AI*. 459–465.

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

Selman, B. and Kautz, H. 1993. Domain-independent extensions to GSAT. Technical report, AI Principles Research, AT & T Bell Laboratories, Murray Hill, NJ.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A New Method for Solving Hard Satisfiability Problems. In *Proc. 10th National Conference on AI*. 440–446.

Sosič, R. and Gu, J. 1991. Fast search algorithms for the N-queens problem. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6):1572–1576.