Simulated Annealing for Hard Satisfiability Problems†

William M. Spears
AI Center
Naval Research Laboratory
Washington, D.C. 20375-5320
E-mail: SPEARS@AIC.NRL.NAVY.MIL

Abstract

Satisfiability (SAT) refers to the task of finding a truth assignment that makes an arbitrary boolean expression true. This paper compares a simulated annealing algorithm (SASAT) with GSAT (Selman *et al.*, 1992), a greedy algorithm for solving satisfiability problems. GSAT can solve problem instances that are extremely difficult for traditional satisfiability algorithms. Results suggest that SASAT scales up better as the number of variables increases, solving at least as many hard SAT problems with less effort. The paper then presents an ablation study that helps to explain the relative advantage of SASAT over GSAT. Finally, an improvement to the basic SASAT algorithm is examined, based on a random walk suggested by Selman *et al.* (1993).

1 Introduction

Satisfiability (SAT) refers to the task of finding a truth assignment that makes an arbitrary boolean expression true. For example, the boolean expression *a & b* is true iff the boolean variables *a* and *b* are true. Satisfiability is of interest to the logic, operations research, and computational complexity communities. Due to the emphasis of the logic community, traditional satisfiability algorithms tend to be sound and complete. However, Selman *et al.* (1992) point out that there exists a class of satisfiability problems that are extremely hard for these algorithms. Their response has been to create a greedy algorithm (GSAT) that is sound, yet incomplete (i.e., there is no guarantee that GSAT will find a satisfying assignment if one exists). The advantage of GSAT is that it can often solve problems that are extremely difficult for the traditional algorithms.

Other recent work has also concentrated on incomplete algorithms for satisfiability (De Jong and Spears, 1989; Spears, 1990; Young and Reel, 1990; Gu, 1992). However,

† This is an expanded version of NCARAI Technical Report #AIC-93-015.

comparisons between the algorithms have been difficult to perform, due to a lack of agreement on what constitutes a reasonable test set of problems. One nice feature of the Selman *et al.* (1992) paper is that a class of hard problems is very precisely defined. In this paper we compare GSAT with a novel simulated annealing approach (SASAT) on that class of hard problems. The results suggest that SASAT solves at least as many problems with much less effort. Next, we modify the simulated annealing algorithm, to illustrate why SASAT outperforms GSAT. Finally, the paper examines an enhancement to SASAT, based on a random walk feature described in Selman *et al.* (1993). First, however, we provide an overview of GSAT and introduce the simulated annealing algorithm.

2 GSAT and SASAT

GSAT assumes that the boolean expressions are in conjunctive normal form (CNF). After generating a random truth assignment, it tries new assignments by flipping the truth assignment of a variable that leads to the largest increase in the number of true clauses. GSAT is greedy because it always tries to increase the number of true clauses. If it is unable to do this it will make a "sideways" move (i.e., change the truth assignment of a variable although the number of true clauses remains constant). GSAT can make a "backwards" move, but only if other moves are not available. Furthermore, it can not make two backwards moves in a row, since the backwards move will guarantee that it is possible to increase the number of true clauses in the next move. The algorithm for GSAT is presented in Figure 1.

```
Procedure GSAT;
Input: A set of clauses, MAX_FLIPS, and MAX_TRIES;
Output: A satisfying truth assignment of the clauses, if found;

 for i = 1 to MAX_TRIES {
   T = a random truth assignment;
   for j = 1 to MAX_FLIPS {
     if T satisfies the clauses then return T;

     flip a variable that results in the largest increase
     δ in the number of clauses made true;

     T = the new assignment after the flip is made;
 } }
```

Figure 1: The GSAT algorithm.

Recently, Spears (1993) showed that a neural network with simulated annealing outperforms GSAT on hard satisfiability problems. The neural network algorithm makes no assumptions about the form of the boolean expression. By specializing to CNF expressions, the neural network can be dropped, resulting in a simulated annealing algorithm we call SASAT. The algorithm for SASAT is presented in Figure 2.

After generating a random truth assignment, SASAT tries new assignments by probabilistically flipping each variable individually, based on the improvement this flip would bring. If the improvement is positive the flip is likely to be performed. If the improvement is negative the flip is unlikely to be performed. SASAT differs from GSAT in that it can make arbitrary sequences of "backwards" moves, which is a necessary feature for escaping local optima in the search space. The search characteristics of SASAT are heavily influenced by the maximum temperature (MAX_TEMP), minimum temperature (MIN_TEMP), and the annealing schedule. The probabilistic moves (flips) are determined using the standard logistic function for simulated annealing. When the temperature is high the moves are almost random, and when the temperature is low SASAT is similar to GSAT. The annealing schedule is calculated using:

$$temperature = MAX\_TEMP * e^{-j * decay\_rate}$$

We used our experience with the neural network algorithm (Spears, 1993) to roughly estimate the temperature parameters. We set MAX_TEMP to 0.3, MIN_TEMP to 0.01, and the decay rate to be:

$$decay\_rate = \frac{1}{i * V}$$

where V is the number of variables and i is a loop variable in the SASAT algorithm. Thus, each time i is incremented the decay rate decreases. Also, the decay rate is dependent on the number of variables in the problem to be solved. SASAT will use smaller decay rates on problems with more variables, because dropping the temperature more slowly is a good heuristic for larger problems.

```
Procedure SASAT;
Input: A set of clauses, MAX_TRIES, MAX_TEMP, and MIN_TEMP;
Output: A satisfying truth assignment of the clauses, if found;

  i = 0; tries = 0;
  loop {
     i++;
     T = a random truth assignment; j = 0;
     loop {
        if T satisfies the clauses then return T;

        temperature = MAX_TEMP * e^(-j * decay_rate);
        if (temperature < MIN_TEMP) then exit loop;

        for v = 1 to the number of variables V in the clauses {
           Compute the increase (decrease) δ in the number of
           clauses made true, if v were flipped;

           flip variable v with probability defined by the
           logistic function: 1 / (1 + e^(-δ/temperature)) ;

           T = the new assignment if the flip is made;
        }
        j++; tries++;
        if (MAX_TRIES = tries) terminate algorithm;
  }  }
```

Figure 2: The SASAT algorithm.

3 Experiments and Results

In comparing two (or more) algorithms, one difficult choice is in the selection of problem instances to solve. Since traditional satisfiability algorithms already work well on many problems, it is useful to consider those problems where the traditional algorithms run into difficulty. Furthermore, to avoid the risk of overfitting an algorithm to a particular problem, it is important to either select a large number of problems or to draw problems from a particular problem class (or distribution). Fortunately, classes of problems that are difficult for traditional satisfiability algorithms have already been identified. In this paper we will concentrate on one such class, a fixed clause-length model referred to as *Random L-SAT* (Mitchell *et al.*, 1992). The performance of GSAT on Random L-SAT problems has already been reported in Selman *et al.* (1992). Although we could not obtain the specific problems used in their experiments, we generated random problems using their random problem generator.

Another difficult choice is in *how* to measure and compare the performance of the two algorithms. Clearly, one important measure is the average amount of computation performed by both algorithms. For both GSAT and SASAT, the bulk of the computation lies in two operations, the computation of the δs and the manipulation of the data structures when the flip is made. Although both a flip and the computation of each δ can be performed fairly efficiently through the use of carefully chosen data structures, the complexity increases with the number of clauses. Other steps in the algorithms, such as calls to a random number generator, the check for termination, the computation of the logistic function, or the determination of the best variable (in GSAT) appear to be less computationally intensive.

Because they are the computationally intensive steps, we will report both the number of δs computed and the number of flips performed by GSAT and SASAT. Selman *et al.* (1992) did not report the number of δs computed by GSAT, but they can be estimated from the reported number of flips. At first blush the number of δs computed would appear to be the number of flips multiplied by the number of variables V, since an obvious way to find the best variable is to compute the δ for each variable, selecting a variable with the highest δ. However, as Selman points out (personal communication), after each flip it is only necessary to compute the δs of those variables that share one or more clauses with the flipped variable.

Random L-SAT problems are in conjunctive normal form, with L literals per clause, and C clauses per variable. Thus a problem contains $L * C * V$ literals and a variable occurs in approximately $L * C$ clauses. From this we can conclude that a given variable shares clauses with at most $L * C * (L - 1)$ variables, which is independent of V. In other words, at most $L * C * (L - 1)$ δs are computed for every flip in GSAT.

One interesting feature of the Random L-SAT problems is that the hardest problems (for traditional algorithms) occur where the clause to variable ratio is roughly 4.25. Furthermore, when C is 4.25, roughly 50% of the random problems appear to be satisfiable.† Since we are generating random problems from this distribution, it is insufficient to simply report the average number of δs and flips required to satisfy those problems that were actually satisfied. This is because different algorithms may actually solve a different percentage of the satisfiable problems. In order to have a more meaningful comparison, then, it is important to report the percentage of problems satisfied, as well as the amount of effort required to satisfy them.

Following Selman *et al.* (1992), we generated random 3-SAT problems ranging from 100 to 500 variables, where C is 4.25. All results are averaged over 100 random

† Crawford and Auton (1993) believe that a better estimate is $4.25\ V + 6.21$.

instances for each choice of the number of variables. We monitored the number of δs computed and flips performed by SASAT, and estimated the number of δs computed by GSAT, based on the results in Selman *et al.* (1992). Since C is 4.25, roughly $3 * 4.25 * 2 = 25.5$ δs are computed for every flip in GSAT, and we will use this result to estimate the number of δs computed by GSAT. We also present the percentage of problems satisfied by SASAT. The percentages of problems satisfied by GSAT are not reported, however Selman (personal communication) states that GSAT satisfies roughly 20% - 33% of the 500 variable problems. We will assume that GSAT satisfies roughly 50% of the easier problems.

Variables	Clauses	δs	Flips	% Solved	MAX_FLIPS†
100	425	541,875	21,250	˜50%	500
200	850	12,673,500	497,000	NR†	2,000
300	1275	35,465,400	1,390,800	NR	6,000
400	1700	89,943,600	3,527,200	NR	8,000
500	2125‡	253,929,000	9,958,000	20-33%	10,000

Table 1: GSAT on hard problems.

Variables	Clauses	δs	Flips	% Solved	MAX_TRIES
100	425	581,400	31,863	58/100	200,000
200	850	7,735,000	396,341	44/100	400,000
300	1275	37,474,500	1,924,040	48/100	800,000
400	1700	42,951,600	2,269,500	45/100	1,000,000
500	2125	86,680,500	4,438,820	41/100	1,600,000

Table 2: SASAT on hard problems.

Tables 1 and 2 present the percentage of problems actually satisfied by the algorithms. They also give the number of δs computed and flips performed, averaged over the problems that were satisfied. Given all this data, how do we compare the two algorithms? Should we use flips, δs, or some combination of the two? Fortunately we can finesse this decision. As mentioned before, due to clever design, GSAT computes roughly 25.5 δs for each flip, for this particular class of problems. Interestingly, if we compare the ratio of δs to flips in SASAT we see a similar pattern. For all choices of the number of variables, the ratio is roughly 20. Although not a result of design, this rather fortuitous ratio allows us to assume that both GSAT and SASAT do roughly the same amount of work per flip. For this reason we will concentrate on comparing the two algorithms on the percentage of problems satisfied and the average number of flips required to satisfy those problems.

Figure 3 graphs the number of flips for both algorithms. Thus, in terms of flips, although GSAT may have some advantage on the easier problems, SASAT appears to

† NR means that this datum has not been reported. MAX_TRIES is also not reported, but is at least 10(Flips / MAX_FLIPS).

‡ Selman *et al.* (1992) report in their table that they used 2150 clauses, yet they also state that they use $4.25V$ clauses on the harder problems. We follow the $4.25V$ guideline.

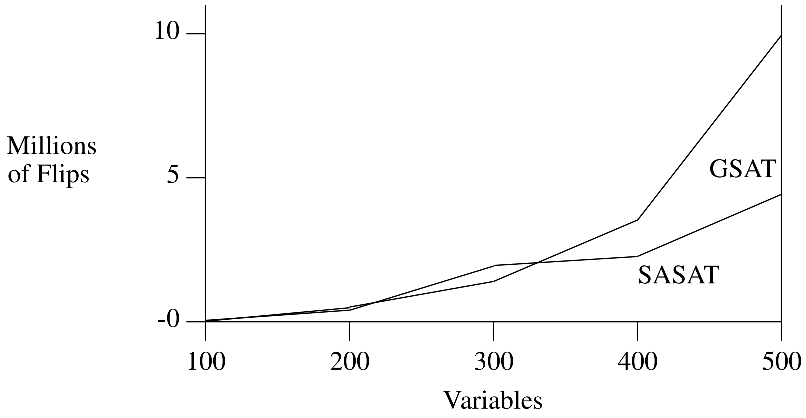

Figure 3: Comparison of GSAT and SASAT.

scale better. A comparison on percentages is harder. The percentage of problems solved by GSAT was not strictly monitored. Also, when the number of variables were high the results for GSAT were averaged over only 10 satisfied problems. However, in general SASAT appears to solve a higher percentage of problems with fewer flips.

3.1 Distributions of Results

One of the difficulties in dealing with the above 400 and 500 variable problems is that there are no known techniques for practically determining which of the problem instances are in fact satisfiable. Thus, the results given above could be sensitive to the choices of cutoffs for the algorithms. For example GSAT is run with specific choices for the cutoff parameters MAX_TRIES and MAX_FLIPS. SASAT is run with specific choices for the parameters MAX_TEMP, MIN_TEMP, and MAX_FLIPS. With the parameters set as given above, SASAT appears to solve a higher percentage of problems with less work. However, it is unlikely that SASAT satisfied all the satisfiable problems. Thus, it is conceivable that the situation could reverse if the cutoffs were increased to such an extent that both algorithms solved more (or all) of the problems. In other words, the remaining unsatisfied but satisfiable problems *could* be much more difficult for SASAT then GSAT.

Thus, if we don't know which problems are satisfiable, conclusions will necessarily be tentative. An alternative would be to solve only problems with less than 300 variables, since the satisfiable problems can be determined by sound and complete techniques. Unfortunately, our experience has shown that is also hard to draw any conclusions about the behavior of algorithms at 500 variables from their behavior at 300 variables. In either case we would have to draw tentative conclusions.

Although it is impossible to firmly resolve this issue with the current state of sound and complete algorithms, we can provide the distribution of problems actually satisfied by SASAT. This distribution graphs the number of problems solved within a certain amount of work. Although not a complete distribution (i.e., it is very likely to not include some satisfiable problems), this distribution will be helpful in comparing other algorithms with SASAT in the future. Before providing the distributions, however, it is instructional to consider the variance of the results given above. Table 3 provides the standard deviation of the flips for SASAT.

Variables	Clauses	Flips	Standard Dev.
100	425	31,863	88,117
200	850	396,341	795,558
300	1275	1,942,040	2,911,738
400	1700	2,269,500	3,718,480
500	2125	4,438,820	7,970,972

Table 3: Standard deviation of flips for SASAT.

In each case the standard deviation is higher than the mean, indicating the presence of outliers in the data. In other words, we will expect that the distribution will contain a small number of problems that are much more difficult than average. In order to confirm this we graphed the distribution - showing the number of satisfied problems solved within a certain number of flips. Figures 4, 5, and 6 give the distributions for the 100, 300, and 500 variable problems.

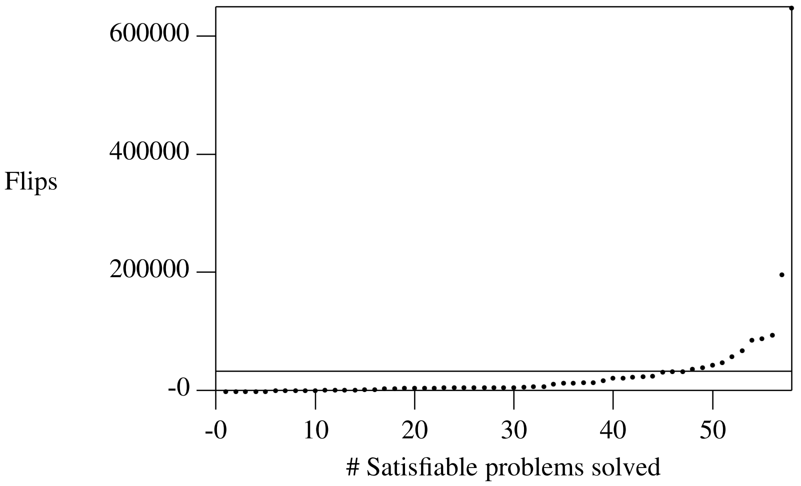

Figure 4: Distribution for SASAT on 100 variable problems.

In Figures 4, 5, and 6 we represent the mean number of flips by a solid horizontal line. As expected, the majority of the problems were solved with less than the mean number of flips, and the presence of the outliers dramatically increases both the mean and the standard deviation. This agrees with conventional wisdom, which states that most NP-Complete problem classes are dominated by a small number of truly difficult problems.

One interesting use of these distributions is in deciding whether an unsatisfied problem is satisfiable or not. If there is any reason to believe that a problem has been drawn from a distribution similar to a Random L-SAT distribution, and SASAT has been attempting to satisfy that problem without success, Bayesian analysis can be used to estimate the probability that the problem is in fact unsatisfiable. Details of this technique can be found in Spears (1992).

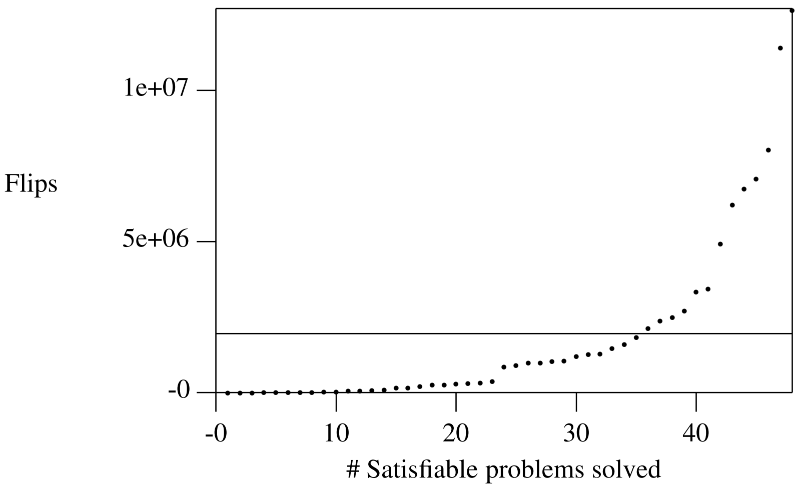

Figure 5: Distribution for SASAT on 300 variable problems.

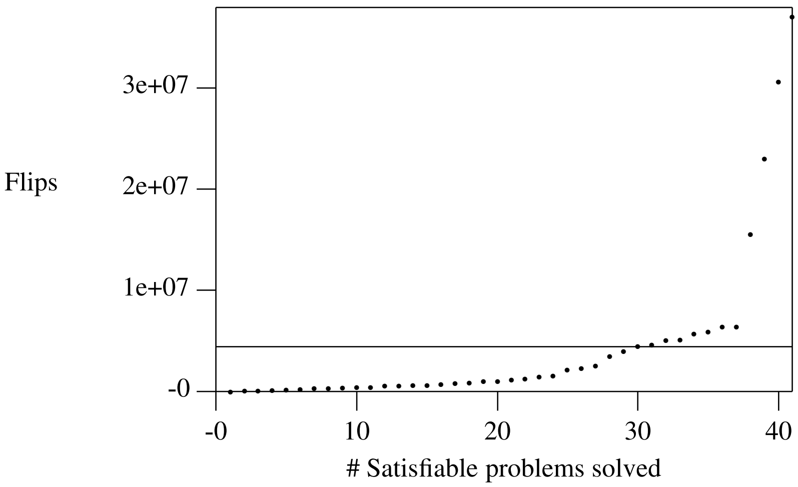

Figure 6: Distribution for SASAT on 500 variable problems.

4 A Modification to SASAT

As stated above, comparisons of SASAT with GSAT are at best difficult to make. Selman also has reported comparisons of GSAT with simulated annealing. The results have been mixed (Selman and Kautz, 1993; Selman *et al.* 1993). Given the difficulty in making a comparison it is reasonable to wonder if SASAT is really doing better than GSAT, and if so, why? To help answer these questions we modified SASAT to make it more similar to GSAT. This is easily done by using a zero temperature logistic function, that never makes a backwards move (see Figure 7), Although SASAT would still not be choosing the *best* variable to flip, it would nevertheless only make sideways or forwards moves, as GSAT primarily does.

flip variable v with probability defined by the
logistic function:

```
if (δ < 0) return 0.0;
else if (δ == 0) return 0.5;
else return 1.0;
```

Figure 7. Zero temperature logistic function for SASAT.

The motivation behind this modification is the idea that backwards moves are the primary advantage of SASAT over GSAT. If this is true, we would expect the zero temperature SASAT to perform more like GSAT, both in terms of the percentage of problems satisfied, and the amount of work required to satisfy them. In order to test this hypothesis we reran the above experiments using the zero temperature SASAT. Table 4 provides the results.

Variables	Clauses	δs	Flips	% Solved	MAX_TRIES
100	425	422,000	19,853	54/100	200,000
200	850	8,518,000	376,200	51/100	400,000
300	1275	43,578,900	1,947,180	38/100	800,000
400	1700	99,440,400	4,469,170	28/100	1,000,000
500	2125	246,889,500	11,019,800	23/100	1,600,000

Table 4: Zero temperature SASAT on hard problems.

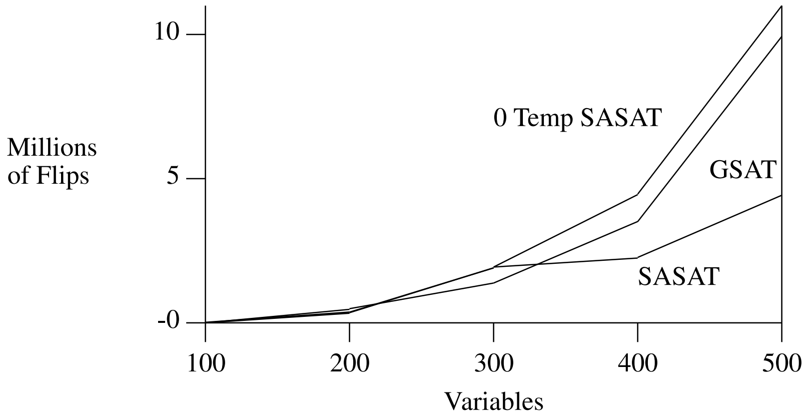

Figure 8: Comparison of GSAT, SASAT, and zero temperature SASAT.

Again, we compare on both the percentages and the number of flips (Figure 8). Tables 1 and 4 suggest that the percentages of problems satisfied by the zero temperature SASAT fall within the values estimated for GSAT. Furthermore, Figure 8 suggest that

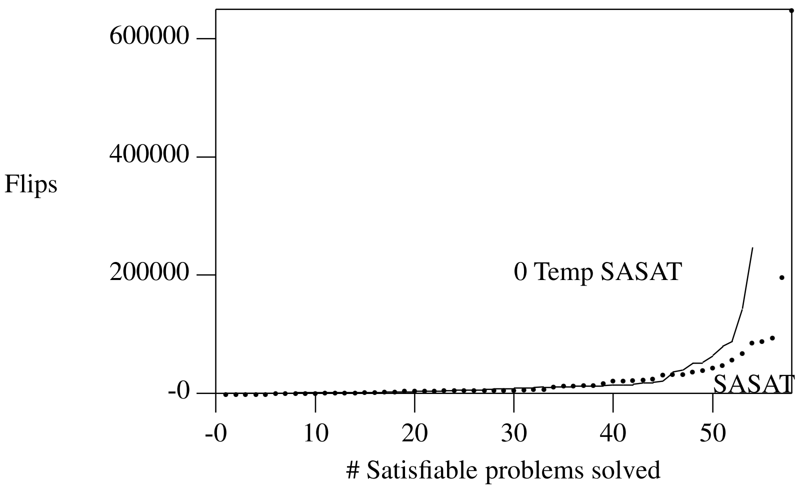

Figure 9: Distribution for zero temperature SASAT on 100 variable problems.

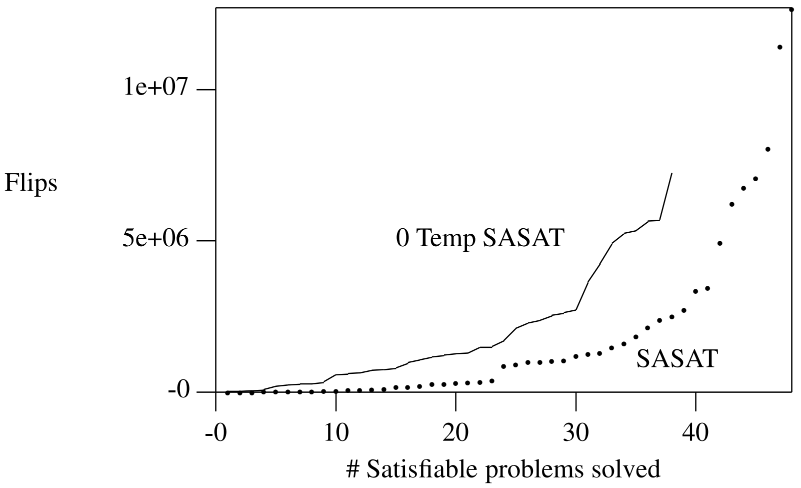

Figure 10: Distribution for zero temperature SASAT on 300 variable problems.

GSAT and zero temperature SASAT scale similarly. As expected, the zero temperature SASAT algorithm appears to behave very much like GSAT. Figures 9, 10, and 11 compare the distributions of the zero temperature SASAT with SASAT. The distribution of the modified SASAT is presented as a solid line. The distributions indicate that the relative advantage of SASAT increases as the number of variables increase.

These results highlight a number of interesting points. First, as expected, the zero temperature performs similarly to GSAT. Second, since SASAT outperforms zero temperature SASAT, we strengthen our conclusion that SASAT does indeed outperform GSAT on this class of problems, at least with the given cutoffs. Third, the annealing schedule is the key to the relative advantage of SASAT. A good annealing schedule

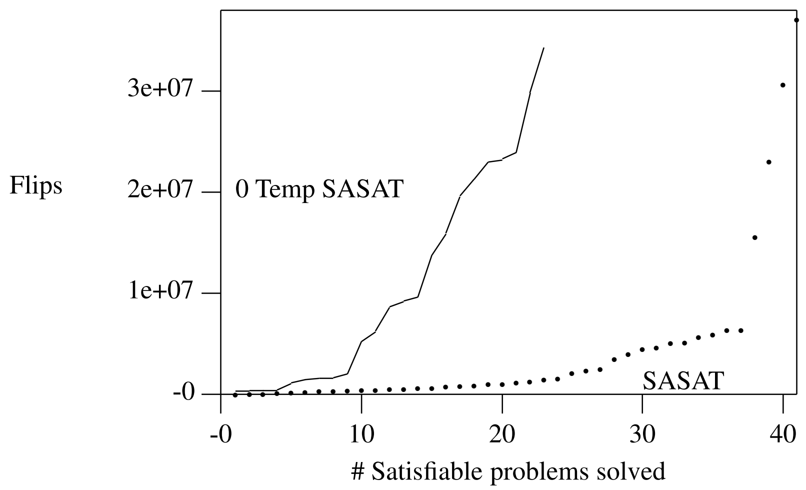

Figure 11: Distribution for zero temperature SASAT on 500 variable problems.

allows SASAT to escape suboptima sufficiently often that SASAT outperforms zero temperature SASAT (and GSAT) both in the percentage of problems satisfied and the number of flips required to satisfy them. Finally, the actual annealing schedule used helps explains the mixed results reported by Selman. In Selman's experiments relatively high maximum temperatures were used - namely, 5 or 10. SASAT has a maximum temperature of 0.3. Clearly temperatures much higher than 0.3 simply result in a lot of wasted search.

Despite the modification to SASAT, GSAT remains different in one other important aspect. Unlike SASAT, GSAT always flips the *best* variable of the *V* variables than can be flipped at any time. In other words GSAT bases its decisions on the global information obtained from all variables. SASAT, on the other hand, bases its decisions only on the local information associated with one variable. Interestingly, it is not clear that the global mechanism is very useful, since zero temperature SASAT (which uses local information) performs in a fashion that is very similar to GSAT. This raises an intriguing question. Is choosing the *best* really a good strategy? We plan to pursue this in more detail in the future.

5 SASAT with a Random Walk

As mentioned earlier, one characteristic of SASAT is that it allows arbitrary sequences of backwards moves. Recently, GSAT has been enhanced by a feature referred to as a random walk (Selman *et al.*, 1993). The purpose of the walk is to allow GSAT to escape from local optima by making backwards moves. However, the random walk moves are more specific than those made by SASAT. Periodically, GSAT randomly chooses an unsatisfied clause, and flips the value of a random literal within that clause (thus making that clause true). Preliminary results indicate that this is an effective heuristic for GSAT.

Considering the relative advantages that SASAT appears to have, it is reasonable to also consider adding a similar heuristic to SASAT. One elegant way is to modify the logistic function as shown in Figure 12.

flip variable v with probability defined by the logistic function:

```
with probability p {
  if (v is in an unsatisfied clause) return 1.0;
  else return 0.0;
}
else with probability 1 − p {
  return 1 / (1 + e^(−δ/temperature));
}
```

$$\text{return } \frac{1}{1 + e^{-\frac{\delta}{temperature}}};$$

Figure 12. Random walk logistic function for SASAT.

Thus, with probability p we check to see if a variable is in an unsatisfied clause. If it is we flip it. If not, we leave it alone. Finally, with probability $1 - p$ we use the standard logistic function. The motivation is to add backwards moves to SASAT that are not simply random. In fact, random walk moves are targeted towards those clauses that appear to be giving the algorithm difficulty.

Of course, the behavior of this algorithm depends greatly on the value chosen for p. Results from the genetic algorithm community suggest that such perturbations should occur roughly once for each pass over the V variables. Drawing on these results, and some preliminary empirical experiments, we set p to $1/V$ and reran SASAT. Table 5 presents the results.

Variables	Clauses	δs	Flips	% Solved	MAX_TRIES
100	425	102,900	8,072	55/100	200,000
200	850	1,833,400	125,239	56/100	400,000
300	1275	19,202,100	1,236,040	53/100	800,000
400	1700	34,179,200	2,128,179	46/100	1,000,000
500	2125	99,556,000	6,016,920	46/100	1,600,000

Table 5: SASAT with random walk on hard problems.

Again we compare the algorithms on the percentages of problems solved, and the number of flips required to solve them (Figure 13). SASAT with the random walk solves a higher percentage of problems than it did before, achieving almost 50%. In terms of the average number of flips, it does not appear to scale as well as SASAT. However, this is somewhat misleading, since SASAT with the random walk is solving more problems. Figures 14, 15, and 16 compare the distributions of SASAT with the random walk against SASAT. The distribution for SASAT with the random walk is presented as a solid line. Note that, with the possible exception at 100 variables, SASAT with the random walk appears to be a definite improvement over the basic SASAT algorithm.

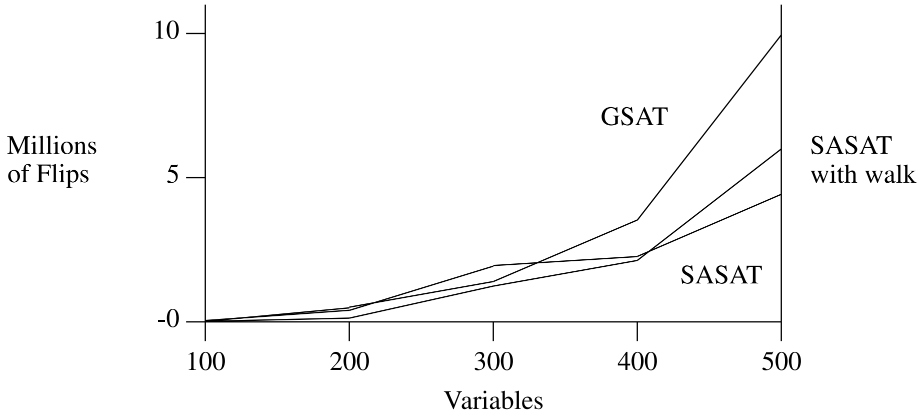

Figure 13: Comparison of GSAT, SASAT, and SASAT with random walk.

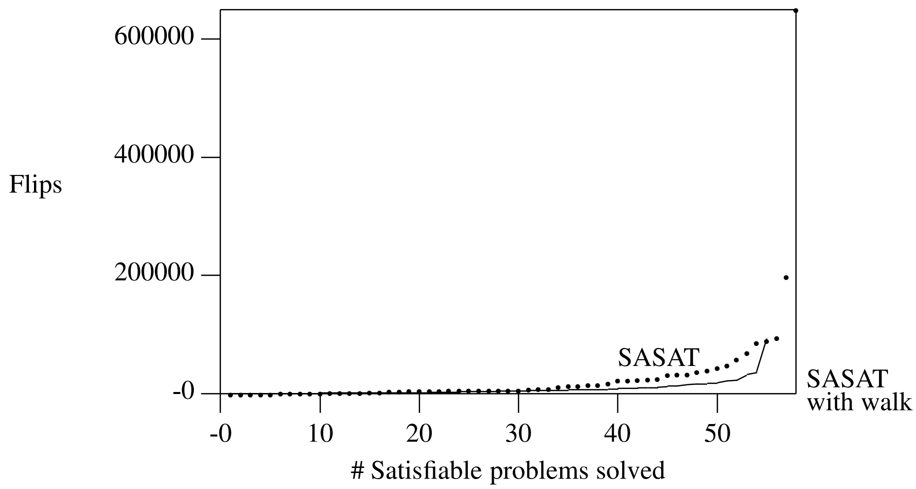

Figure 14: Distribution for SASAT with walk on 100 variable problems.

6 Summary and Discussion

In this paper we considered an application of simulated annealing (SASAT) to a class of hard problems and compared the resulting algorithm with a greedy algorithm (GSAT). With the given cutoffs, SASAT appears to satisfy at least as many hard SAT problems as GSAT, with less work. We then presented evidence confirming that the relative advantage of SASAT lies in its use of random backward moves. Finally, by adding a random walk heuristic, SASAT was shown to solve even more problems.

There are a number of important issues that still need to be addressed. First, the conclusions reached in this paper are necessarily tentative, given that it was impossible to know which unsatisfied problem instances were in fact satisfiable. One resolution to

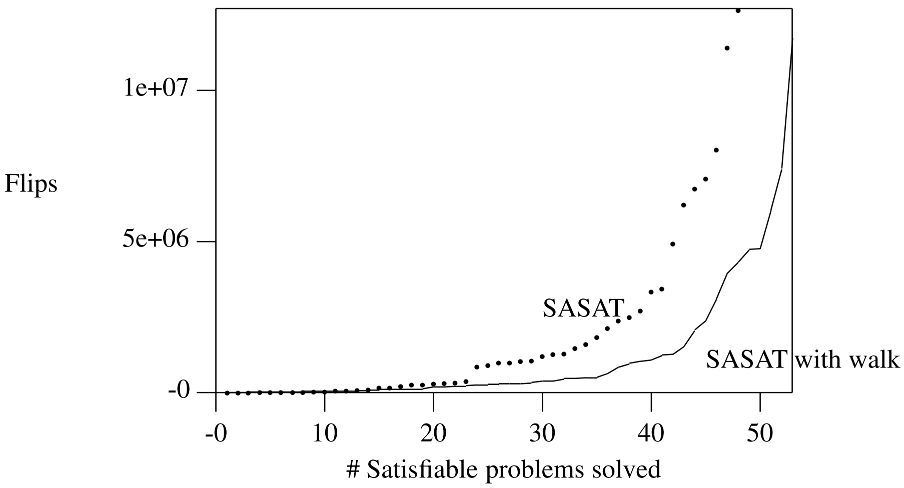

Figure 15: Distribution for SASAT with walk on 300 variable problems.

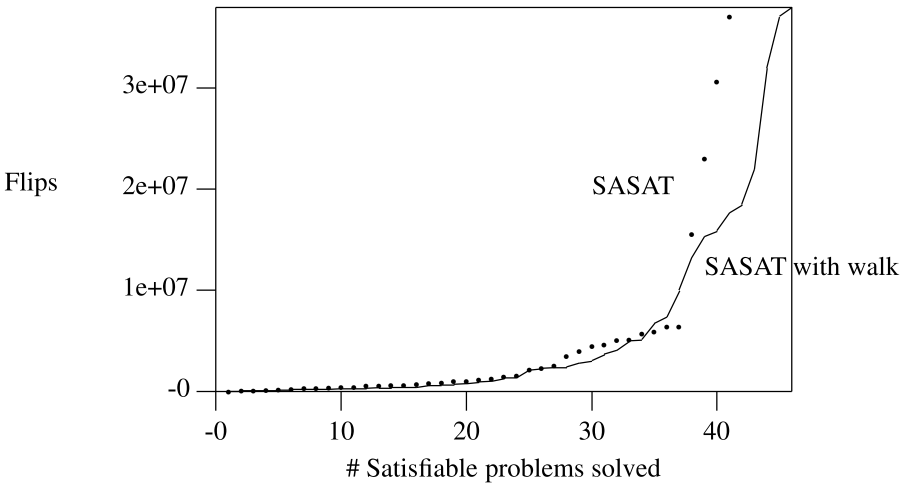

Figure 16: Distribution for SASAT with walk on 500 variable problems.

this problem will be to improve our sound and complete algorithms, allowing us to know which problems are satisfiable and unsatisfiable. However, at some point the problems will become too difficult for the sound and complete algorithms. At this point the only way to ultimately test the incomplete algorithms is to simply run them on the harder problems and display as much of the problem distribution as is possible.

Second, the issue of fairness with respect to comparisons is still open. Given the current implementations of SASAT and GSAT, and the problem class explored, flips appears to be a good measure for comparison. For Random 3-SAT problems where C is 4.25, GSAT computes roughly 25 δs per flip. Fortuitously SASAT computes roughly 20 δs per flip. However, SASAT with random walk computes less than 16 δs per flip. Is the comparison still fair? Furthermore, on Random L-SAT problems where *L* is greater than

3, GSAT will compute more than 25 δs per flip. It is unknown how SASAT will behave.

To a large extent, the good performance of GSAT rides on the observation that it is only necessary to compute the δs of those variables that share one or more clauses with the last flipped variable. We have not taken advantage of this observation in SASAT. For example, it is highly unlikely that SASAT *needs* to recompute the 20 δs that occur (on the average) between flips. If this is true, it will be possible to reduce the amount of work required per flip in SASAT. Note that, as with GSAT, this will not change the semantics of the algorithm. Preliminary experiments indicate that, per flip, the number of δ computations can be reduced by roughly 35 percent. We will explore this potential further in the near future.

Finally, it can be argued that the Random L-SAT problems are not of interest, since they may not occur in realistic problems. We will explore the behavior of SASAT on other problem classes (or instances) in future research.

Acknowledgements

Thanks to Diana Gordon and Ken De Jong for many helpful comments on this work and this paper. Diana was also highly influential in my efforts to find a fair comparison between GSAT and SASAT. I would also like to thank David Johnson, Bart Selman, Ian Gent, Toby Walsh, and Antje Beringer for provocative and insightful comments. Any remaining errors are of course the author's responsibility.

References

Crawford, J. M. & L. D. Auton (1993) Experimental Results on the Crossover Point in Satisfiability Problems, *Proceedings of the 1993 AAAI Conference*, Washington, DC, pgs. 21 - 27.

De Jong, K.A. & W. Spears (1989) Using Genetic Algorithms to Solve NP-Complete Problems, *International Conference on Genetic Algorithms*, George Mason University, Fairfax, Virginia, June 1989, pgs. 124 - 132.

Gu, J. (1992) Efficient Local Search for Very Large-Scale Satisfiability Problems, *SIGART Bulletin*, 3(1), January 1992.

Mitchell, D., Selman, B., & H. Levesque (1992) Hard and Easy Distributions of SAT Problems, *Proceedings of the 1992 AAAI Conference*, San Jose, CA, pgs 459 - 465.

Selman, B., Kautz, H. A., & B. Cohen (1993) Local Search Strategies for Satisfiability Testing, paper in preparation for the 2nd DIMACS Challenge, Rutgers University.

Selman, B., & H. A. Kautz (1993) An Empirical Study of Greedy Local Search for Satisfiability Testing, *Proceedings of the 1993 AAAI Conference*, Washington, DC, pgs 46 - 51.

Selman, B., Levesque, H., & D. Mitchell (1992) A New Method for Solving Hard Satisfiability Problems, *Proceedings of the 1992 AAAI Conference*, San Jose, CA, pgs 440 - 446.

Spears, W. M. (1993) A NN Algorithm for Hard Satisfiability problems, Artificial

Intelligence Center Internal Report #AIC-93-014, Naval Research Laboratory, Washington, DC 20375.

Spears, W. M. (1992) Probabilistic Satisfiability, Artificial Intelligence Center Internal Report #AIC-92-026, Naval Research Laboratory, Washington, DC 20375.

Spears, W. M. (1990) Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Problems, *Masters Thesis*, Department of Computer Science, George Mason University, Fairfax, Virginia.

Young, R. A. & A. Reel (1990) A Hybrid Genetic Algorithm for a Logic Problem", *Proceedings of the 9th European Conference on Artificial Intelligence*, pp.744-746, Editor: Aiello, L.C. Publisher: Pitman, London, UK.