The GSAT/SA-Familiy – Relating greedy satisifability testing to simulated annealing

S. Hölldobler and H. Hoos and A. Strohmaier and A. Weiß

Intellektik!

Forschungsbericht AIDA–94–17

Darmstadt,

Fachgebiet Intellektik, Fachbereich Informatik
Technische Hochschule Darmstadt
Alexanderstraße 10
D–64283 Darmstadt
Germany

The GSAT/SA Family*
Relating greedy satisfiability testing to Simulated Annealing

Steffen Hölldobler,[†] Holger Hoos,[‡] Antje Strohmaier,[‡] Andreas Weiß[‡]

November 24, 1994

Abstract

In this paper, we investigate and relate various variants of the greedy satisfiability tester GSAT. We present these algorithms as members of a whole family of algorithms for finding a model for satisfiable propositional logic formulas. In particular, all algorithms can be formulated as instances of the same generic frame GenSAT. Comparing the algorithms, we do not only concentrate on their overall performance, but are also interested in how properties like locality or different kinds of randomness influence the performance. To the end we define a new, theoretically complete instance of GenSAT. This variant can be viewed as a reformulation of simulated annealing (SA) within the GSAT family and, thus, defines a link between GSAT and SA. For most of the algorithms experiments have been performed using very hard, randomly generated propositional logic formulas. The results of these experiments are also reported.

1 Introduction

In 1953, a physical model was described by Metropolis et al simulating the behaviour of a gas in a heat bath which is slowly annealed to obtain an optimal crystalline structure [13]. This model was later called Simulated Annealing (SA) and as one of its main characteristics it is based on local information and a good deal of randomness. In 1982, Kirkpatrick et al investigated that SA could be used to compute approximative solutions for optimization problems like the Traveling Salesman Problem [12]. Then, S. and D. Geman were able to show the theoretical convergence of this algorithm to the set of optimal solutions in 1984 [8]. In terms of theorem proving, SA is complete. However, SA was ignored by the majority of the AI community — above all the researchers interested in theorem proving did not take notice of this algorithm, which was guaranteed to find solutions to their kind of problems only in the limit, i.e. after infinitely long computation time.

In contrast, in 1992 a new method for solving hard satisfiability problems called GSAT (Greedy SATisfiability tester) was presented by Selman et al [22] and almost immediately attracted the attention of many AI researchers. GSAT's decisions are guided by a so–called "greedy" behaviour, i.e. it tries to maximize the locally possible improvement of the current solution. Thereby it also uses some global information. GSAT was found to outperform Davis-Putnam

* This work was supported by DFG Grant HO 1294/3-2.
† Wissensverarbeitung, Informatik, TU Dresden, D-01062 Dresden, Germany, sh@ikid10.inf.tu-dresden.de.
‡ Intellektik, Informatik, TH Darmstadt, Alexanderstraße 10, D-64283 Darmstadt, Germany.

procedures [6] in a region of very hard randomly generated propositional formulas [22] and was able to solve problems that were simply too large for existing decision procedures. In the sequel, starting from the more abstract frame "GenSAT" for GSAT-like procedures, several variants could easily be described and distinguished [10]. On the one hand they were used to investigate the contribution of several GSAT-features to its performance [10] and, on the other hand, further improvements of GSAT could be described easily. In particular, a probabilistic feature called "random walk" was introduced, which improved the performance of GSAT significantly [21].

Some features of this new version of GSAT resemble very much the old–fashioned SA algorithm: its search is conducted by local information and it contains probabilistic elements. But the older optimization method SA has an advantage over GSAT: There exists a theoretical convergence result proving the (theoretical) completeness of SA. Consequently, it can be guaranteed that there is a correlation between the computation time and the quality of the solution found up to this point. Although this result may seem little useful for satisfiability problems where only the optimal solutions (i.e. models of the given propositional formula) are interesting, it can also be used for non–monotonic logics like the penalty logic [17], where a set of contradicting constraints with different penalties is given and the satisfiable subsets with the least total penalty are wanted. Furthermore, experiments with GSAT — which even theoretically is not guaranteed to find a solution — show that very often an optimal solution, i.e. a model for the given formula, is found in practice and, in particular, there is no need for infinitely long computation times. Hence, we expected SA to be as useful in practice as GSAT showed up to be.

So did other people and they compared GSAT and SA. However, these comparisons supported contradicting conclusions. Whereas Selman, Kautz, and Cohen [20] used GSAT with random walk for the comparison, which scales up significantly better than basic GSAT, Spears [24] uses only basic GSAT for the comparison. In addition, [24] uses the results of [23] which report a significantly higher amount of flips per assignment than new experiments with basic GSAT reported in [20], and he compares basic GSAT with a variant of SA performing also random walk, which scales up even better than SA itself. To summarize, [20] see GSAT as the "winner" whereas [24] conjectures that SA scales up better. But both comparisons seemed to be preferential, which is why we started to do our own experiments.

The arising questions were then: How does the GSAT–family look like? What relationships concerning the performance and behaviour of different instances of the GenSAT–frame during the search exist? How is SA related to GSAT? Can we classify SA in terms of the GenSAT–family? Is there a correlation in the behaviour of SA and GSAT?

In the sequel of this paper we will investigate these questions. After some preliminaries presented in Section 2 we will formally define GSAT in Section 3. Abstracting from a description of the original GSAT algorithm, we will then introduce the GenSAT frame due to [10] in Section 4. Thereafter, we will discuss the different features and behaviour of instances of GenSAT. In particular, the probabilistic variant GWSAT, i.e. GSAT with "random walk", [21] will be presented in Section 5. A cautious GenSAT called CSAT [9] and its modifications ISAT [10] and I_2SAT will be discussed in Sections 6, 7, and 8, respectively.

As we will show, the performance of I_2SAT is almost identical to the performance of GSAT although the algorithms differ in one important aspect: Whereas GSAT's decisions are based on a global parameter, I_2SAT's decisions are based on local conditions alone and, hence, I_2SAT may be implemented much more efficiently. On the other hand, I_2SAT's performance will turn out to be still worse than GWSAT's performance. Hence, as a kind of natural step we will combine the locality of I_2SAT with the randomness of GWSAT to obtain a new instance of GenSAT called IWSAT (Section 9). IWSAT will turn out to perform as well as GWSAT. Interestingly, an extension of ISAT with random walk didn't improve the performance as significantly[1].

All GenSAT variants presented so far are theoretically incomplete. In order to tackle the question whether there is a theoretically complete GenSAT variant which performs as good as IWSAT in practice we will abstract from IWSAT to another algorithmic frame PSAT in Section 10. PSAT's decisions are based on a local distribution function and, hence, various instances of PSAT can be obtained by selecting different distribution functions. In Section 11 we will choose the distribution function usually used in SA algorithms to obtain the variant ASAT. We will then be able to adapt the theoretical completeness results for SA to this GenSAT variant (Section 12). Figure 1 gives an overview of the GSAT/SA–family investigated in this paper.

We have tested most of the variants for randomly generated propositional logic formulas in conjunctive normal form. In particular, we have chosen random formulas which are known to be very hard [4, 14], i.e. formulas, where each disjunct contains three literals and the number of clauses m is equal to $4.3 \times N$, where N denotes the number of variables occuring in the formula. For $N = 100$ and $N = 200$ we used 100 formulas and for $N = 300$ and $N = 500$ we used 50 formulas in the experiments. To ensure a fair competition, we have tested each variant with the same set of formulas. Additionally, we have distinguished between so–called forced and unforced formulas. Forced formulas are constructed to be satisfiable, whereas unforced formulas are constructed purely at random and then the unsatisfiable ones are filtered out by means of a complete decision procedure. It will turn out that the various variants behave differently on forced and unforced formulas. Table 1 shows the results of our experiments. These results are discussed in the final Section 13.

2 Preliminaries

Let Σ be a set of (propositional) variables. A *literal* is either a variable $V \in \Sigma$ or the negation $\neg V$ of a variable $V \in \Sigma$. A *clause* is a finite disjunction of literals. A *propositional logic formula in conjunctive normal form* (CNF), or *formula* for short, is a finite conjunction of clauses. Hence, each formula Φ is of the form

$$\Phi \equiv \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} L_{ij},$$

where each L_{ij} denotes a literal.

An *assignment* A for a formula Φ is a total mapping from the set $\mathcal{V}(\Phi)$ of variables occurring in Φ to the truth values $\top$ ($\mathtt{true}$ or 1) and $\bot$ ($\mathtt{false}$ or 0), i.e. $A \in \{0,1\}^{\mathcal{V}(\Phi)}$. The application $\Phi[A]$ of an assignment A to a formula Φ is inductively defined as follows.

- If Φ is a variable then $\Phi[A]$ is the truth value assigned to Φ by A.
- If Φ is of the form $\neg\Phi_1$ then $\Phi[A] = \top$ if $\Phi_1[A] = \bot$, and $\Phi[A] = \bot$, otherwise.
- If Φ is of the form $\Phi_1 \wedge \Phi_2$ then $\Phi[A] = \top$ if $\Phi_1[A] = \Phi_2[A] = \top$, and $\Phi[A] = \bot$, otherwise.

[1] I. P. Gent, personal communication.

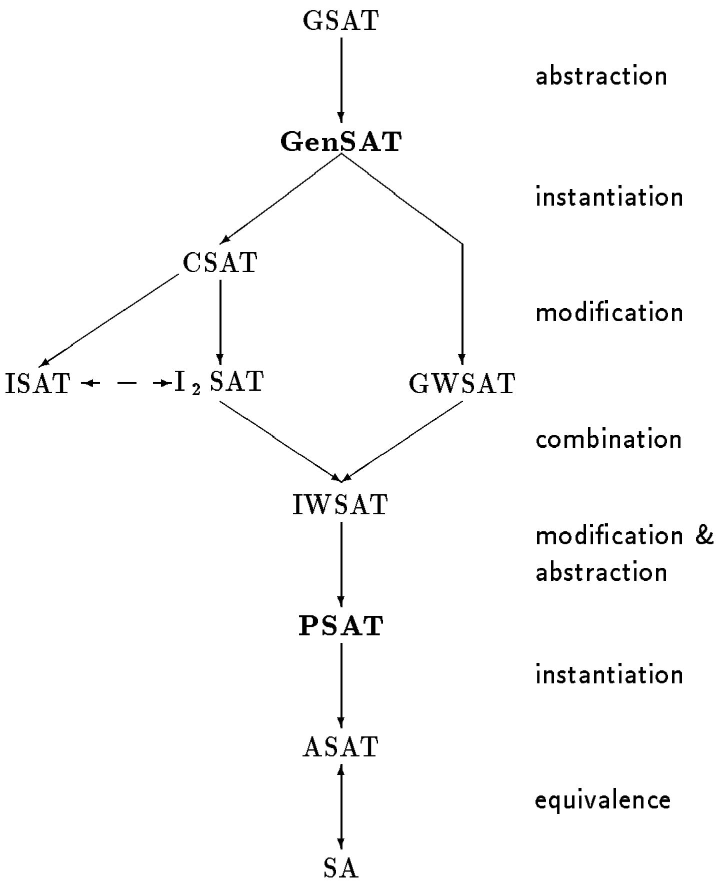

Figure 1: The GSAT/SA Family. Abstracting from GSAT the generic frame GenSAT can be used to obtain all other variants as instances of GenSAT. The dashed line between ISAT and I_2SAT indicates, that both variants are based on the same idea although there are some slight variations, which nevertheless lead to some significant difference in the performance of ISAT and I_2SAT. PSAT is also an instance of GenSAT, but it is a generic frame as well as it is based on a local distribution function which must be further instantiated. ASAT, a reformulation of SA, is such an instance of PSAT.

N	variant	forced		unforced	
		assign	F/A	assign	F/A
100	GSAT	76.1%	3911	23.5%	34120
100	I_2SAT	75.2%	4087	23.9%	33398
100	GWSAT	100%	520	82.2%	4776
100	IWSAT	100%	554	82.6%	4720
100	ASAT 0.7	99.8%	607	70%	6649
100	ASAT rnd	98.2%	1243	69.3%	6800
200	GSAT	75.3%	9597	8.4%	223750
200	I_2SAT	74.5%	9792	8.7%	216341
200	GWSAT	99%	1622	48.7%	28695
200	IWSAT	99.9%	1636	45.7%	31784
200	ASAT 0.7	99%	2422	35.9%	43841
200	ASAT rnd	96.8%	4482	23.1%	76182
300	GSAT	77.0%	15360	2.7%	1091611
300	I_2SAT	74.2%	17118	2.3%	1288396
300	GWSAT	100%	2462	26.3%	98059
300	IWSAT	99.8%	2586	21.3%	125623
300	ASAT 0.7	99.4%	4699	16.3%	168352
300	ASAT rnd	96.6%	8564	7.6%	382693
500	GSAT	70.2%	34830	–	–
500	I_2SAT	73.8%	31849	–	–
500	GWSAT	100%	4731	–	–
500	IWSAT	100%	5016	–	–
500	ASAT 0.7	95.4%	14968	–	–
500	ASAT rnd	86.8%	26838	–	–

Table 1: The results of our experiments with GSAT, GWSAT, I_2SAT, IWSAT, and two ASAT variants. We have run these experiments for formulas Φ with $N = 100$, 200, 300, and 500 variables. The column labeled "assign" shows the percentage of tries in which an assignment satisfying Φ was found and the column labeled "F/A" shows the average number of flips necessary to find one assignment, where MAXTRIES $= 1$. The notions "flips", "MAXTRIES", etc. are defined in Sections 2 and 3. "–" indicates that no experiments could be performed as the unforced formulas where too large for the decision algorithm needed to filter out the unsatisfiable formulas.

- If Φ is of the form $\Phi_1 \vee \Phi_2$ then $\Phi[A] = \bot$ if $\Phi_1[A] = \Phi_2[A] = \bot$, and $\Phi[A] = \top$, otherwise.

A is said to be a *model* for Φ iff $\Phi[A] = \top$.

In the sequel let Φ be a formula and A an assignment. $\mathcal{V}(X)$ denotes the set of variables occurring in the syntactical object X. $\mathcal{C}(\Phi)$ denotes the set of clauses occurring in Φ. $\mathcal{C}_u(\Phi, A) = \{c \in \mathcal{C}(\Phi) \mid c[A] = \bot\}$ is the set of clauses occurring in Φ which are unsatisfied under A. $A[V]$ denotes an assignment which is equal to A except that the truth assignment of the variable V is flipped. More precisely, $V[A[W]] = \neg V[A]$ if $W = V$, and $V[A[W]] = V[A]$ if $W \neq V$. The *score* of a variable V wrt Φ and A is the difference between the number of clauses left unsatisfied by A and the number of clauses left unsatisfied by $A[V]$, i.e. $score(\Phi, A, V) = |\mathcal{C}_u(\Phi, A)| - |\mathcal{C}_u(\Phi, A[V])|$. For example, a positive score represents a decrease, whereas a negative score represents an increase in the number of unsatisfied clauses caused by the flip of V. The computation of the score of V wrt A is called an *update* of V.

An *energy function* is a function $E : \{0,1\}^n \to \mathbb{N}_0$. Let A be an assignment. The application $E[A]$ of A to the energy function E is the natural number obtained by replacing each occurrence of a variable V in E by the value assigned to V by A. For each formula Φ an energy function E can be constructed such that the variables occurring in E and Φ are identical and for each assignment A the value of $E[A]$ is equal to $|\mathcal{C}_u(\Phi, A)|$ (see e.g. [17]). More precisely, for a formula

$$\Phi \equiv \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} L_{ij}$$

the corresponding energy function is

$$E_\Phi = \sum_{i=1}^{m} \prod_{j=1}^{n_i} E_{L_{ij}},$$

where

$$E_{L_{ij}} = \begin{cases} X & \text{if } L_{ij} \equiv \neg X \\ 1 - X & \text{if } L_{ij} \equiv X \end{cases}$$

For example, the energy function $E = 1 - A + AB - BC$ corresponds to the formula $\Phi \equiv (A \vee B) \wedge (\neg B \vee C)$. If Φ is satisfiable, then its models correspond to the global minima of E, which are the assignments A with $E[A] = 0$. If Φ is unsatisfiable, then the global minima of the corresponding energy function have a value greater than 0. Hence, searching for a model of a formula Φ is nothing but searching for a global minimum in the landscape defined by the corresponding energy function E. This is also the reason why GSAT and its variants are incomplete: If the algorithms terminate in a minimum of the energy function which has a value greater than 0, then we do not know whether this is a non–global minimum for a satisfiable formula or the global minimum for an unsatisfiable one.

In order to explain the behaviour of the following search procedures we use an informal way to describe the energy landscape. Actually, the landscape consists of discrete points over the n-dimensional hypercube. We nevertheless use a continuous imagination. The *neighbourhood* of an assignment A is the set

$$N_A = \{B \in \{0,1\}^{\mathcal{V}(\Phi)} | \exists V \in \mathcal{V}(\Phi) : B = A[V].\}$$

In other words, the neighbours of an assignment are those assignments with a Hamming distance of 1. One *step* means moving from the current assignment to one of its neighbours by flipping a single variable.

A set $\mathcal{A}$ of variable assignments is called *connected* if

$$\forall A \in \mathcal{A}: \ \exists B \in \mathcal{A}, \ V \in \mathcal{V}(\Phi): \ A = B[V].$$

A connected set of variable assignments $\mathcal{A}$ defines a *plateau* $x \in \mathbb{N}_0$ on the energy function E if

$$\forall A \in \mathcal{A}: \ E[A] = x.$$

A connected set of variable assignments $\mathcal{A}$ defines the *top of a hill* if

$$\forall B \notin \mathcal{A}, \ A \in \mathcal{A}, \ V \in \mathcal{V}(\Phi): \ B = A[V] \Longrightarrow E[B] < E[A].$$

3 GSAT

The GSAT procedure is a local greedy search strategy operating on a propositional formula Φ in conjunctive normal form [22]. Informally, GSAT starts with a random truth assignment A and, while no satisfying assignment is found, it flips a variable V with the highest score. Therefore, GSAT is called greedy. If after MAXFLIPS steps still no satisfying assignment is found, GSAT chooses a new random assignment. This is repeated for at most MAXTRIES times (see Figure 2).

```
procedure GSAT (Φ) % Φ is a propositional formula in CNF
  for i := 1 to MAXTRIES do
    A := random assignment for variables in Φ ;
    for j := 1 to MAXFLIPS do
      if Φ[A] = ⊤ then return A ; end if;
      V := a variable with highest score wrt Φ and A ;
      A := A[V] ;
    end for;
  end for;
  return "no satisfying assignment found";
end GSAT;
```

Figure 2: The GSAT Procedure.

Usually, GSAT and its variants are run with MAXTRIES set to some $n \geq 1$. As the purpose of this paper is to compare various variants of GSAT, it suffices to run these variants with MAXTRIES $= 1$. The performance of these variants with MAXTRIES $= n$ can be computed statistically as follows. Let N denote the fact that GSAT terminates successfully with MAXTRIES set to n and let S_1 denote that GSAT terminates successfully with MAXTRIES set to 1. One should observe, that each of the MAXTRIES attempts to find a model is independent. Consequently, the probability $P(S_n)$ that GSAT finds a model with MAXTRIES $= n$ can be obtained as $P(S_n) = 1 - P(\neg S_1)^n$. For instance, as shown in Table 1 GWSAT and IWSAT solve 26.3%

and 21.3% of the unforced formulas with 300 variables, respectively, where MAXTRIES is set to 1. With MAXTRIES set to 15 we find that GWSAT and IWSAT solve 98.9% and 97.2%, respectively. So in practical applications, differences in performance are less significant than they appear to be.

4 GenSAT

Besides the basic GSAT depicted in Figure 2 a variety of different versions of this algorithm have been investigated. To relate these versions Gent and Walsh abstracted away from the basic GSAT and defined a generic frame called GenSAT [10]. Figure 3 shows a slightly modified frame.

```
procedure GenSAT (Φ)
  for i := 1 to MAXTRIES do
    A := initial(Φ);
    for j := 1 to MAXFLIPS do
      if Φ[A] = ⊤ then return A; end if;
      possflips := hillClimb(Φ, A, j);
      if possflips ≠ ∅ then
        V := pick(possflips);
        A := A[V];
      end if
    end for;
  end for;
  return "no satisfying assignment found";
end GenSAT;
```

Figure 3: The GenSat Frame.

Here the three function parameters are introduced to describe more generally the characterisitc features of GSAT-like procedures. These are *initial*, *hillClimb*, and *pick*. The function *initial* always chooses a random assignment and *pick* always picks an arbitrary variable from the given set of variables (see Figure 4). These functions will not be changed throughout the paper. The function *hillClimb* determines the set of possible flips with the help of a *select* function, which selects candidate–variables with respect to the given function Φ, the current assignment A, and some other parameters. The GenSAT instances considered herein mostly differ in their definition of the *select* function. The generic *select* function has a number of formal parameters. To obtain a uniform description every specific *select* function declares all of these, even if some are not used in a given context. In the parameter list, Φ denotes the formula, A the current assignment, V the variable that is to be checked, and j the number of the current flip. The last parameter, r, is a random number between 0 and 1 on whose value some *select* functions choose between alternative patterns of behavior (e.g. GWSAT or IWSAT).

We modified the GenSAT frame introduced in [10] by inserting a test whether the set of possible flips is empty. Such a test has no effect on the versions discussed in [10], viz. on GSAT, CSAT, ISAT and GWSAT (see Figure 1), as in these versions the set of possible flips cannot

```
procedure initial(Φ)
  return random assignment for variables in Φ ;
end initial;

procedure pick(possflips)
  return randomly selected element of possflips;
end pick;

procedure hillClimb(Φ, A, j)
  r := rnd(0,1);      % random number between 0 and 1
  return {V | select(Φ, A, V, j, r) = true} ;
end hillClimb;
```

Figure 4: The *initial*, *pick*, and *hillClimb* Procedures.

be empty. There is also no effect on I_2SAT and IWSAT. All these variants do exactly one flip per execution of the inner loop of the algorithm. By the way, this is the reason for calling the corresponding parameter MAXFLIPS. But for the procedures PSAT and ASAT the set of possible flips may be empty.

As mentioned before, the various GSAT–versions are obtained from the GenSAT frame by instantiating the *select* function. For example, the *select* function shown in Figure 5 instantiates the GenSAT frame to become the basic GSAT procedure. As other GSAT–versions differ in the *select* function, we added an appropriate index to this function.

$$select_{GSAT}(\Phi, A, V, j, r) = \begin{cases} \texttt{true} & \texttt{if } score(\Phi, A, V) = max\{score(\Phi, A, W) \mid W \in \mathcal{V}(\Phi)\}, \\ \texttt{false} & \texttt{otherwise}. \end{cases}$$

Figure 5: The *select* Function for GSAT.

In the sequel we discuss some experimental results of the performance of various instances of GenSAT summarized in Table 1. If not stated otherwise, we always use the following parameter setting:

$$\text{MAXTRIES} = 1,$$
$$\text{MAXFLIPS} = 100 \times N\,.$$

5 GWSAT — GSAT with Random Walk

GWSAT is an instance of GenSAT, whose performance is significantly superior to the one of basic GSAT [21]. It was designed because, roughly speaking, basic GSAT always falls in the next local minimum and practically has no means to escape from it besides restarting the search at another random point on the energy surface. GWSAT includes another feature to escape local minima called *random walk*: With a certain fixed probability p the set `possflips` consists of all variables occurring in yet unsatisfied clauses (see Figure 6). In other words, even variables with non–maximal score may be selected for flipping. This includes also variables with negative score. Hence, the variable selected from this set may cause an *upwards* move in the energy landscape. Besides restarting the search by choosing a new random assignment this is a further method to escape from local minima as it allows to escape at least from narrow valleys.

$$select_{GWSAT}(\Phi, A, V, j, r) = \begin{cases} \texttt{true} & \texttt{if } r \geq p \texttt{ and} \\ & score(\Phi, A, V) = max\{score(\Phi, A, W) \mid W \in \mathcal{V}(\Phi)\}, \\ \texttt{true} & \texttt{if } r < p \texttt{ and } V \in \mathcal{V}(\mathcal{C}_u(\Phi, A)), \\ \texttt{false} & \texttt{otherwise}. \end{cases}$$

Figure 6: The *select* Function for GWSAT.

The results of our experiments comparing basic GSAT and GWSAT are given in Table 1. Obviously, the forced formulas are significantly easier than the unforced ones as the success rates are always higher. If we compare the success rates of basic GSAT, we can furthermore state that the energy functions of forced formulas have less local minima compared to the number of global minima than those of unforced formulas: Starting at a random point in the energy landscape, basic GSAT goes straight to the next minimum without effective uphill moves.[2] Hence, it's success rate essentially gives the portion of global minima in the total number of minima. For instance in the 100 variable problems, 76.1% of the minima of forced formulas are global ones whereas only 23.5% of the minima of unforced formulas are global ones.

6 CSAT — A cautious GenSAT–Variant

CSAT is an instance of GenSAT first introduced by [9] to investigate if greediness is important for the performance of GSAT–style procedures. This method is less greedy than basic GSAT. It shows indifferent behaviour towards the selection of variables for `possflips` as it does not distinguish between improving flips of varying score. In other words, all variables with positive score are selected. If such variables do not exist, then all variables with score 0 are selected or, if this set is also empty, then all variables — which all have negative score — are selected (see Figure 7). As shown by [9] the performance of CSAT is not significantly worse than that of GSAT.

[2] It may move upwards if all scores are negative and let us assume, it did so by flipping the variable V. Then the next move will be downwards as there is now at least one positive score, viz. by flipping V again. Hence, GSAT has a very limited capability of escaping from local minima. It may escape if the minimum consists of a single point in the energy landscape and one of its neighboring points is the top of a hill — and this is the only kind of minima GSAT can escape from.

$$select_{CSAT}(\Phi, A, V, j, r) = \begin{cases} \texttt{true} & \texttt{if } score(\Phi, A, V) > 0 \\ \texttt{true} & \{W \mid score(\Phi, A, W) > 0\} = \emptyset \texttt{ and } score(\Phi, A, V) = 0 \\ \texttt{true} & \{W \mid score(\Phi, A, W) \geq 0\} = \emptyset \\ \texttt{false} & \texttt{otherwise.} \end{cases}$$

Figure 7: The *select* Function for CSAT.

7 ISAT — An Indifferent GenSAT–Variant

A modification of CSAT is ISAT, which was introduced in [10] and which is indifferent to downwards and sideways flips. It distinguishes only between flips with score greater or equal to 0 and with score less than 0. In other words, it combines the first two cases of the selection function for CSAT (see Figure 8). This variant may be implemented much more efficiently than basic GSAT as it is unnecessary to update all scores per step in order to determine the current maximal score. As soon as a variable with score greater or equal 0 is found it can be picked. But, unfortunately, as shown in [10] ISAT performes significantly worse than basic GSAT.

$$select_{ISAT}(\Phi, A, V, j, r) = \begin{cases} \texttt{true} & \texttt{if } score(\Phi, A, V) \geq 0 \\ \texttt{true} & \texttt{if } \{W \mid score(\Phi, A, W) \geq 0\} = \emptyset \\ \texttt{false} & \texttt{otherwise.} \end{cases}$$

Figure 8: The *select* Function for ISAT.

8 I_2SAT — Another Indifferent GenSAT–Variant

Following a slightly different approach we obtain another indifferent modification of CSAT which we call I_2SAT. The *hillClimb* function of basic GSAT returns the set of all flips with maximal score. Hence, its selection function is based on a global decision which makes parallelization difficult. From this perspective it would be nice to have a GenSAT–variant which does not need this global information while not performing significantly worse than basic GSAT. The most obvious way to achieve this is to include all improving flips, i.e. flips with strictly positive score, into the set `possflips`. If no such flips exist, the set of all possible flips will be returned (see Figure 9). This is exactly the definition of I_2SAT, which in all other respects is identical to basic GSAT. This variant is indifferent to sideways and upwards steps—it distinguishes only between improving and non–improving flips. In other words, the second and third case of CSAT's selection function are combined.

The motivation which led from basic GSAT to ISAT was to investigate the question, whether

$$select_{I_2SAT}(\Phi, A, V, j, r) = \begin{cases} \texttt{true} & \texttt{if } score(\Phi, A, V) > 0 \\ \texttt{true} & \texttt{if } \{W \mid score(\Phi, A, W) > 0\} = \emptyset, \\ \texttt{false} & \texttt{otherwise.} \end{cases}$$

Figure 9: The *select* Function for I_2SAT.

greediness is an important feature of GSAT, wheras I_2SAT is motivated by the wish to eliminate GSAT's need for global decisions. Actually, ISAT as well as I_2SAT constructs **possflips** by local decisions only. This property can be used to increase the performance of the algorithms in the sequential and parallel case. The basic idea is to randomly select variables within *hillclimb* and to return the first variable with positive score in the case of I_2SAT (or non–negative score in the case of ISAT). Figure 10 shows such an improved *hillclimb* procedure, where it is no longer necessary for the appropriate *select* functions to determine $\{W \mid score(\Phi, A, W) \rhd 0\}$, $\rhd \in \{>, \geq\}$: Only if this set is really empty, the **for** loop is executed completely and then all variables will be returned. To guarantee fair selection, the variables have to be checked in random order. This is important, because Gent and Walsh have discovered that in GenSAT–variants the violation of such a fairness condition causes significant decrease in performance [10].

```
procedure hillClimb(Φ, A, j)
  P  := rndPerm(𝒱);       % random permutation of N variable-indices
  r  := rnd(0,1);         % random number between 0 and 1
  for k := 1 to N do
    if select(Φ, A, P_k, j, r) then return({P_k}); % kth element of P
  end for;
  return (𝒱);
end hillClimb
```

Figure 10: An Improved *hillClimb* Function for ISAT and I_2SAT.

According to our experiments, I_2SAT does not perform significantly worse than basic GSAT (see Table 1). One should observe that the kind of indifference realized by I_2SAT is quite different from the one we find in ISAT, although both algorithms appear to be almost identical. Whereas I_2SAT treats equally all improving flips, ISAT even makes no difference between downward and sideward moves. Consequently, the behaviour of both algorithms in the plateau phase of the search, i.e. the phase where flips do not change the energy, is different. Whereas ISAT will be wandering around in a plateau until it hazardously finds another variable causing a downwards step, I_2SAT may move upward in this case as all variables are included in **possflips** as soon as no improving flips exist. Consequently, I_2SAT is not only more greedy but also better in escaping from local minima than ISAT. It is more greedy because it always chooses an improving flip if this is possible. It is better in escaping from local minima if the local minima are not single points but plateaus. This may explain the result of [10] that ISAT does indeed perform significantly worse than GSAT, which stands in contrast to our results comparing GSAT with I_2SAT (see Table 1).

9 IWSAT

Given the significant improvement in performance achieved by the random walk concept and the advantage of a local selection function for *hillClimb* realized by I_2SAT, the kind of natural step is to combine these features. This way we obtain a new GenSAT–variant, which we call IWSAT. In this algorithm every time *hillClimb* is called, it decides whether to behave exactly like I_2SAT's *hillClimb* or to perform a step of random walk in the style of GWSAT. This decision is made on a random variable r and a threshold p as in GWSAT (see Figure 11).

$$select_{IWSAT}(\Phi, A, V, j, r) = \begin{cases} \texttt{true} & \texttt{if } r \geq p \texttt{ and } (score(\Phi, A, V) > 0 \texttt{ or} \\ & \{W \mid score(\Phi, A, W) > 0\} = \emptyset) \qquad \texttt{\% I}_2\texttt{SAT--part} \\ \texttt{true} & \texttt{if } r < p \texttt{ and } V \in \mathcal{V}(\mathcal{C}_u(\Phi, A)), \qquad \texttt{\% GWSAT--part} \\ \texttt{false} & \texttt{otherwise.} \end{cases}$$

Figure 11: The *select* Function for IWSAT.

In our experiments we found IWSAT's performance being not significantly worse than GWSAT's, a fact which is not very surprising given the analogous results for I_2SAT and basic GSAT. Our results are also given in Table 1. One should observe that the algorithms are compared by the number of flips needed to find an assignment. These results do not reflect the time needed to perform a flip. As mentioned in Section 8, I_2SAT may use an improved *hillClimb* procedure and, thus, we expect that a flip computed by I_2SAT using such a procedure takes less time than a flip in GWSAT which cannot use the improved *hillClimb*.

10 PSAT

Although variants of GSAT like GWSAT and IWSAT are quite impressive in practice they are less impressive in theory. Theoretical results about the behaviour of these algorithms during the search for a model are difficult to obtain. They are incomplete — and this holds even if MAXFLIPS is allowed to increase, which led to a much better performance in our experiments. These difficulties seem to be based on the problem of how to handle the set $\mathcal{C}_u(\Phi, A)$ of unsatisfied clauses occurring in a formula Φ under an assignment A theoretically.

Our experiments indicate that there is a link between the number of unsatisfied clauses and the score of a variable wrt a given formula and an assignment. Unfortunately, however, we were unable to find a formal correlation between these two values. On the other hand, GWSAT as well as IWSAT have probabilistic features. Can we use or extend these features such that a probabilistic variant of GSAT shows a similar behaviour in practice while allowing to make formal statements about the behaviour during the search for a model and about the completeness of the algorithm? Our basic idea explored in the sequel is to use an appropriate probability distribution for the selection of variables which is based on the score of a variable. More precisely, we will only define a probability for the selection of a variable.

$$P(\mathit{select}(\Phi, A, V, j, r)) = f(\mathit{score}(\Phi, A, V), j),$$

where f is a local distribution function, i.e. a function which can be computed locally, and $\mathit{select}(\Phi, A, V, j, r)$ is a discrete stochastic variable[3]. This variable has only two possible values, viz. **true** and **false**. Therefore, the distribution function is sufficiently defined, as $P(\neg\mathit{select}(\Phi, A, V, j, r)) = 1 - f(\mathit{score}(\Phi, A, V), j)$.

Now, PSAT is a variant of GenSAT where the selection of variables is based on such a probability distribution (see Figure 12). Similarly to ISAT and I_2SAT an improved *hillClimb* function may be applied. A set of at most one variable is returned, if the randomly chosen variable is selected (see Figure 13).

$$\mathit{select}_{PSAT}(\Phi, A, V, j, r) = \begin{cases} \texttt{true} & \texttt{if } r \leq f(\mathit{score}(\Phi, A, V), j) \\ \texttt{false} & \texttt{otherwise.} \end{cases}$$

Figure 12: The *select* Function for PSAT.

```
procedure hillClimb(Φ, A, j)
   V  := pick(𝒱);
   r  := rnd(0,1);
   if select(Φ, A, V, j, r) then return({V});
   else return (∅);
end hillClimb
```

Figure 13: The *hillClimb* Function for PSAT. $\mathtt{rnd}(0,1)$ yields a random number between 0 and 1.

One should observe, that PSAT is itself a frame as the function f is not given and, thus, PSAT can be instantiated by specifying f. In the following Section 11 such an instantiation is given.

There is another important difference between PSAT and the variants of GSAT considered so far. Whereas the *hillClimb* function used by these variants always returns a non-empty

[3] Please note, that the probability of the *select* function to have the value **true** does not depend on the parameter r as may be suggested by our notation. The parameter r is only used to realize the probability distribution of *select* and is a random number drawn each time, *select* is called See also Figure 13

set of variables, this set may be empty in the case of PSAT. Consequently, PSAT's inner loop may be executed without flipping a variable. Thus, the running variable j may be incremented although no variable has been flipped and, as a consequence, the loop may be terminated without MAXFLIPS flips being performed.

11 ASAT

There is only a very small step from PSAT to ASAT, a formulation of Simulated Annealing (SA) within the GenSAT frame. We only have to instantiate the function f appropriately (see Figure 14). In ASAT, the probability of upwards moves depends directly on the scores. In addition, a parameter T called (pseudo–) temperature is used to control the influence of the score on the flip probability. If T is rather high, then the influence of the score is weak: For T near infinity the flip probability of each variable is about 1. For T near 0 the flip probability is strongly dependent on the score: If $score(V) > 0$ the flip is always accepted, whereas if $score(V) < 0$ the flip is almost always rejected. The probability for flips with score 0 can be defined arbitrarily without affecting the theoretical properties of the algorithm. In our case it is also equal to 1. By choosing appropriate values for T we may therefore additionally control the probability of upwards moves as it can be done by choosing p in IWSAT. Furthermore, during the execution of ASAT the temperature is changed by means of a so–called annealing schedule such that the influence of the score becomes higher by the time. The problem is to find a "good" approximation of the infinitesimally slow annealing required for the theoretical proof that ASAT finds a global minimum (as given in the following section). The annealing schedule usually depends on the number of iterations j through the inner loop of the algorithm and a certain maximal difference ΔT between two consecutive values of the temperature. It is necessary to find a compromise between very slow annealing (which improves the probability that a global minimum will be found), and keeping the time for the search short. In practice, the temperature is set to a predefined maximal value *MaxTemp* and then annealed until it reaches a given minimal value *MinTemp*. Beyond this point, gradient descent is performed (which is equivalent to setting $T(j) = 0$) until an energy minimum is reached. There are many possible annealing schedules. Some examples are

$$\begin{array}{ll}
\text{Schedule } 1: & T(t+1) = T(t) - \Delta T \\
\text{Schedule } 2: & T(t+1) = T(t) - \Delta T / t \\
\text{Schedule } 3: & T(t+1) = \mathit{MaxTemp}/(t+1) \\
\text{Schedule } 4: & T(t+1) = \mathit{MaxTemp}/\exp(t * \Delta T) \\
\text{Schedule } 5: & T(t+1) = T(t) - \Delta T / \exp(t)
\end{array}$$

where t depends on j, e.g. $t = j$ div n. Due to the huge variety of possible parameter settings for the annealing schedule, it is very difficult to find a "good" one for a whole class of problems. We tried several schedules and finally found the following to perform best in our experiments:

$$\begin{array}{lll}
\mathit{MaxTemp} & := & 0.3 \\
\mathit{MinTemp} & := & 0.01 \\
\Delta T & := & 0.01 \\
T(j) & := & \mathit{MaxTemp} - \frac{\Delta T}{j \,\mathrm{div}\, N}
\end{array}$$

Although we tried very many parameter settings, we were not able to reach the results of IWSAT or GWSAT. ASAT was always worse than these two variants, indicating that the not

$$f(score(\Phi, A, V), j, r) \quad = \quad \begin{cases} 1 & \texttt{if } score(\Phi, A, V) \geq 0 \\ \exp(\frac{score(\Phi, A, V)}{T(j)}) & \texttt{if } score(\Phi, A, V) < 0 \end{cases}$$

Figure 14: The probability distribution for ASAT.

purely random upwards moves guided by unsatisfied clauses are a better heuristic.[4] As the choice of the random number generator and the selection method for variables to be updated appeared to influence the performance of the algorithm significantly (up to 10% less success rate with a "bad" random number generator), we also tried another method of selecting variables to be updated: All variables are visited in sequential order and updated with a certain probability u. So in each run through the variables about $u \times N$ of them are updated and possibly flipped. In our experiments, this selection with $u = 0.7$ was superior to purely random selection. The experimental results are given in Table 1 where the latter selection method is referred to as "ASAT 0.7".

12 ASAT is Complete

The algorithm for ASAT given here can be directly related to the simulated annealing algorithm given in [1]. We claim that it realizes the behaviour described there to prove the asymptotic convergence of SA to the set of optimal configurations (i.e. satisfying assignments) of the problem at hand. Thus, the proof of the theoretical completeness of SA given there can be used also for ASAT.

The behaviour of the ASAT algorithm can be described by means of the theory of Markov chains. A *Markov chain* is a sequence of trials (in our case the assignments), where the probability of the outcome of a given trial (i.e. the new assignment) depends only on the outcome of the previous trial (i.e. the current assignment). A Markov chain is called *inhomogeneous* if the associated transition probabilities of the outcomes depend also on the trial number j within the GenSAT-frame. If the transition probabilities are independent of the trial number, the Markov chain is called *homogeneous*.

Due to [1], we then need a process generating a sequence of homogeneous Markov chains of finite length at descending values of the temperature $T(j)$ (as does the algorithm given in [1]) or, equivalently, one inhomogeneous Markov chain of infinite length (which is obtained by concatenating the homogeneous chains). The latter is realized by ASAT if we choose MAXFLIPS $= \infty$: It is sufficient to have a concatenation of infinitely many homogeneous Markov chains of length equal to the size of the neighbourhood of each possible configuration ($= N$, the number of variables, in our case). We therefore have to decrease the temperature every N steps. This is realized in ASAT by means of the chosen function to compute the current temperature.

[4] Recall that Table 1 shows the results for MAXTRIES $= 1$. If we choose MAXTRIES $= 10$ the differences become quite small (see Section 3 for further details).

Furthermore, the temperature must follow the condition

$$T(j,i) \geq \frac{(N+1)\Delta}{\log(j \operatorname{div} N + 2)} \tag{1}$$

where Δ is the maximal score of a variable, i.e. the maximal energy difference between two neighbouring configurations. This may be assured by choosing an appropriate factor in the cooling schedule and by taking $\Delta = c$ (the number of clauses in the formula, as this is the highest possible energy value). Then the convergence of the inhomogeneous Markov chain to the set of optimal configurations can be shown (Theorem 3.6 in [1]). The conditions for the generation and acceptance as well as transition probabilities of new configurations necessary for this proof are naturally fulfilled in the setting considered here for ASAT. So if we use a parameter setting for the annealing schedule satisfying Condition 1, this leads to the following

Proposition 12.1 *If an appropriate annealing schedule is used, ASAT is complete.*

Remember that the meaning of MAXFLIPS is changed in ASAT: it counts rather the updates than flips of variables as executions of the inner loop may occur, where the selected variable is not accepted to flip. Note, that by using MAXFLIPS $< \infty$ and/or a cooling schedule not satisfying condition 1 the theoretical completeness of ASAT is lost. Nevertheless, [1] give polynomial time cooling schedules which do a good approximation of the optimal solution. Furthermore remember that actually, SA is only suited for finding near optimal solutions as finding the exact solution needs computing time at least quadratic in the solution space (such that enumerating and testing all possible configurations would be faster)! But experimental results (as in the GSAT case) nevertheless give good results. Interestingly, also in GSAT increasing the number of MAXFLIPS seems to results in a continuous increase of the success rate. This further shows the similarities between GSAT and SA, which actually needs an infinite number of executions of the inner loop of the algorithm to be complete.

13 Conclusion

The heuristics employed by GSAT–like procedures for finding a model for satisfiable propositional logic formulas has proved to be quite successful. It basically consists of guessing an initial variable assignment followed by flipping a sequence of variables in this assignment. If after a fixed number of flips the initial assignment has not been turned into a model then a new initial variable assignment is selected. The various GSAT–variants differ in the kind of information used to select the variable to be flipped.

In its original version, GSAT selected only variables with maximal score [22] and, therefore, global information was needed. However, as shown in [10], the performance of GSAT — i.e. the number of flips needed before a model is found — does not hinge on the selection of variables with maximal score. Rather it suffices to prefer variables which improve the score and in doing so, the algorithm does not need global information. On the other hand, the performance depends on the behaviour on a plateau. Whereas Gent and Walsh's ISAT performed rather badly, our variant I_2SAT developed in this paper behaves as well as GSAT. Whereas ISAT is wandering around in a plateau, I_2SAT allows upward moves as soon as a plateau is encountered — and this makes the difference. Both, ISAT and I_2SAT, are more local than GSAT, which makes them better candidates for an efficient sequential implementation as they eventually need less time to perform a flip. They are also better candidates for a parallel implementation as locality implies less communication and more independence.

But GSAT was not only refined to be more local, it was also refined to employ random walks [21], and it turned out that GWSAT performed significantly better than GSAT. GWSAT still relies on global information. Our variant IWSAT developed in this paper combines the randomness of GWSAT with the locality of I_2SAT. As IWSAT is more local than GWSAT it is a better candidate for a sequential as well as a parallel implementation for the reasons mentioned at the end of the previous paragraph.

The algorithms of the GSAT family mentioned so far are without exception incomplete. To develop this family further and to obtain an at least theoretically complete GSAT variant, we decided to replace the concept of a random walk by the concept of basing the selection of a variable on a local distribution function. Thereby, we could define a variant ASAT, which essentially is a reformulation of simulated annealing (SA) and, hence, is theoretically complete. In contrast to our expectations SA performed slightly but constantly worse than IWSAT and GWSAT. It is not obvious why this is the case. The cause may be that GWSAT randomly selects only variables which appear in yet unsatisfied clauses, whereas SA may randomly select a variable which only occurs in yet satisfied clauses. This needs to be checked in the future. On the other hand, SA is purely local, whereas IWSAT and GWSAT need some global information. Whether this favours SA as it makes an implementation easier remains also to be seen. On the other hand, there are only two parameters to play with by IWSAT and GWSAT–like algorithms, viz. MAXFLIPS and MAXTRIES , whereas SA–like algorithms always require to find a "good" annealing schedule which may be quite a hassle.

All our developments were checked experimentally. We paid attention to compare the various variants as fair as possible. In particular, we have applied each GSAT variant to the very same randomly generated propositional logic formulas. Thereby it become apparent that there is a difference between forced and unforced formulas, i.e. between formulas which were designed to be satisfiable and formulas which were completely arbitrary and where the satisfiability was checked beforehand. The forced formulas are strictly easier to solve as they appear to have much more global minima than unforced formulas.

Interestingly, the SA variant used in [2] compared well to GWSAT when applied to forced formulas. As mentioned before, SA–like algorithms require to specify an annealing schedule. The schedule in [2] was selected to handle forced formulas well, whereas the schedule in this paper was selected to handle unforced formulas. This may explain the difference between the results on forced formulas reported in [2] and herein. Also, the SA experiments in [2] were performed on the connectionist simulator ICSIM [18], whereas herein we have used an ASAT algorithm implemented in C, which was adapted to the GSAT code we have obtained from Bart Selman . On the specification level the algorithms are equal, but the implementations differ. It is unclear, whether this contributes to the different performances.

The best algorithms of the GSAT/SA family, viz. IWSAT, GWSAT, and ASAT, can eventually be further improved by a variety of techniques. For example, local minima could be filled up using the techniques described in [15, 19, 5], and this might improve the performance of the algorithms, if many local minima exist. If the algorithms are run with MAXTRIES > 1 then knowledge gathered in the first tries could be used to improve the heuristics employed in subsequent tries [10, 19, 16]. Moreover, we may run the algorithm in parallel. There are at least two different ways for doing so. Either apply the random competition paradigm [7, 11] or apply a parallel SA algorithm [3]. During random competition several algorithm are run in parallel and completely independent. Each algorithm uses a different initial assignment and different random number generators. As soon as one algorithm finds a solution the whole computation stops. It appears that even with a rather low number of processors running in parallel, say 10 to 20 , the success rate will be close to 98% for almost all formulas considered herein, i.e. formulas with up to 500 different variables. If a parallel SA algorithm is used, in each step a fixed portion of the variables may be updated and accepted to flip in parallel; the resulting "multiflip" then may be accepted itself depending on a criterion similar to those for the single variable flips. The parallel algorithm given in [3] controlling the multiflips in such a way shows very good results, in contrast to the so-called error algorithm which accepts every multiflip without distinction. It remains to be compared to the sequential algorithms discussed in this paper.

Acknowledgements: We like to thank Gerd Aschemann and Olaf Steinmann for many fruitful discussions, Axel Grossmann for careful reading the final version, and Bart Selman for providing us with the source code of GSAT, a formula generator and a decision procedure for SAT problems.

References

[1] Emile Aarts and Jan Korst. *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons, 1989.

[2] A. Beringer, G. Aschemann, H. Hoos, M. Metzger, and A. Weiß. GSAT versus Simulated Annealing. In A. G. Cohn, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 130–134. John Wiley & Sons, 1994.

[3] N. Boissin and J.-L. Lutton. A parallel simulated annealing algorithm. *Parallel Computing*, 19:859–872, 1993.

[4] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 21–27. MIT press, 1993.

[5] A. Davenport, E. Tsang, Ch. J. Wang, and K. Zhu. GENET: A connectionist architecture for solving constraint satisfaction problems by iterative improvement. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, volume 1, pages 325–330, 1994.

[6] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[7] Wolfgang Ertel. Random competition: A simple, but efficient method for parallelizing inference systems. In B. Fronhoefer and G. Wrightson, editors, *Parallelization in Inference systems*, pages 195–209. Springer, 1992.

[8] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[9] I. P. Gent and T. Walsh. The enigma of SAT hill–climbing procedures. Technical report, University of Edinburgh, 1992. Research Paper 605.

[10] I. P. Gent and T. Walsh. Towards an understanding of hill–climbing procedures for SAT. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 28–33. MIT press, 1993.

[11] Tad Hogg and Colin P. Williams. Expected gains from parallelizing constraint solving for hard problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, volume 1, pages 331–336, 1994.

[12] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Technical Report Research Report RC 9335, IBM Thomas j. Watson Center, Yorktown Heights, N.Y., 1982.

[13] W. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of states calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[14] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 459–465. MIT press, 1992.

[15] P. Morris. The breakout method for escaping from local minima. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 40–45. MIT press, 1993.

[16] Y. Nakakuki and N. Sadeh. Increasing the efficiency of Simulated Annealing search by learning to recognize (un)promising runs. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, volume 2, pages 1316–1322, 1994.

[17] G. Pinkas. *Logical Inference in Symmetric Connectionist Networks*. PhD thesis, Washington University, Sever Institute of Technology, St. Louis, Missouri, 1992.

[18] H. W. Schmidt. ICSIM: Initial design of an object-oriented net simulator. Technical Report TR-90-055, International Computer Science Institute, Berkeley, CA, 1990.

[19] B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In R. Bajcsy, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 1, pages 290–295, 1993.

[20] B. Selman, H. A. Kautz, and B. Cohen. Local search strategies for satisfiabiliy testing. Technical report, AT&T Bell Laboratories, 1993.

[21] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, volume 1, pages 337–343, 1994.

[22] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 440–446, 1992.

[23] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 440–446. MIT press, 1992.

[24] W. M. Spears. Simulated annealing for hard satisfiability problems. Technical report, Naval Research Laboratory, Washington D.C., 1993.