Angewandte Mathematik und
Informatik
Universität zu Köln

Report No. 96.235

# On the Imbalance of Distributions of Solutions of CNF–Formulas and its Impact on Satisfiability Solvers

by
Ewald Speckenmeyer, Max Böhm, Peter Heusch

1996

submitted to AMS/DIMACS series

Universität zu Köln
Institut für Informatik
Pohligstr. 1
D-50969 Köln

**1991 Mathematics Subject Classification:** 68Q22, 68Q25, 68T20
**Keywords:** satisfiability solving, distribution of solutions

# On the Imbalance of Distributions of Solutions of CNF–Formulas and its Impact on Satisfiability Solvers

Ewald Speckenmeyer, Max Böhm, Peter Heusch
Universität zu Köln, Institut für Informatik, Pohligstr. 1, D–50969 Köln
e–mail: {speckenmeyer|boehm|heusch}@informatik.uni–koeln.de

**Abstract**

Let $F$ be Boolean formulas in conjunctive normal form with $n$ variables, $r$ clauses, every clause has length $s$. We show that if $F$ is split into two subformulas $F_v$ and $F_{\overline{v}}$ by setting $v$ true and false in $F$, then the expected number of solutions of one of the two subformulas $F_v$ and $F_{\overline{v}}$ is significantly higher than that in the other subformula, when dealing with classes of formulas where the great majority of formulas is satisfiable. We discuss practical consequences of this result.

## 1 Introduction

Testing Boolean formulas in conjunctive normal form (CNF) for satisfiability is known to be a very time consuming problem, in general. The NP–completeness of the satisfiability (SAT)–problem, see [4], leaves little hope that this situation might change some day. Therefore, when faced with the SAT–problem, we may exploit the special structure of the formula to be solved, like 2–CNF [7], or extended Horn [8], or we may use randomized algorithms if we know in advance that the formula under consideration contains many solutions, see [6, 9], in order to get a quick answer to the problem. But often we have no idea whether we can advantageously exploit the structure of an instance to be solved. In this case we have to resort to a general algorithm like the wellknown Davis Putnam Procedure (DPP). All formulas $F$ considered in this paper are in CNF with $r$ clauses, $F = c_1 \wedge c_2 \wedge \ldots \wedge c_r$. A clause $c = (x_1 \vee \ldots \vee x_s)$ is a disjunction of $s$ literals and a literal $x$ is either a Boolean variable $v$ or its negation $\overline{v}$. $v$ belongs to a set $V$ of $n$ Boolean variables. Denote by $cl(n,s)$ the set of clauses of length $s$ over $V$ and by $cl(n,s)^r$ the space of CNF–formulas with $r$ clauses from $cl(n,s)$. A clause of length 1 is called a **unit clause** and a literal $x$ occuring in $F$ such that $\overline{x}$ doesn't occur in $F$ is called a **pure literal**. The Davis Putnam Procedure (DPP) then tests $F$ for satisfiability as given below. Most implementations of good algorithms for solving CNF formulas are of DPP-type, see [2]. The implementations differ in the data structures for representing formulas, as well as in the selection heuristic for choosing a literal $x$.

The question for a convenient choice of a variable $x$ according to which $F$ is split into $F_x$ and $F_{\overline{x}}$, depends upon whether we expect $F$ to be satisfiable or not.

**procedure** DPP ($F$: CNF-formula)
**begin**
push $F$ onto the stack;
**repeat**
pop top formula $F$ from the stack;
**while** $F$ contains a unit clause $c = (x)$ but no two
complementary unit clauses $c_1 = (x)$ and $c_2 = (\overline{x})$ **do**
choose a unit clause $c = (x)$ and set $F := \{c - \{\overline{x}\} : c \in F, x \notin c\}$
**end;**
**while** $F$ contains a pure literal $x$ **do**
$F := \{c : c \in F, x \notin c\}$
**end;**
**if** empty $(F)$ **then**
report "$F$ satisfiable" and exit
**else if** $F$ contains no complementary unit clauses or the empty clause **then**
select a literal $x \in F$ according to some heuristic;
set $F_{\overline{x}} := \{c - \{x\} : c \in F, \overline{x} \notin c\}$;
push $F_{\overline{x}}$;
set $F_x = \{c - \{\overline{x}\} : c \in F, x \notin c\}$;
push $F_x$
**endif**
**until** stack is empty;
return "$F$ unsatisfiable";
**end;**

In case of unsatisfiability of $F$, $x$ should be chosen such that $F_x$ as well as $F_{\overline{x}}$ are significantly smaller than $F$, because in this case we have to visit the search tree corresponding to $F$ completely, therefore its size should be as small as possible. For satisfiable formulas $F$, however, it is sufficient to detect a satisfying truth assignment of $F$ and we want to visit as little as necessary of the search tree. In the latter case we would like to determine a literal $x$ , s. t. $F_x$ is satisfiable. For if $F_x$ is satisfiable then the second subformula $F_{\overline{x}}$ will be ignored by DPP. Determining some $x$ with this property is a hard problem, in fact it is NP–hard. For this reason, a good heuristic selection rule for determining $x$, such that $x$ hopefully fulfills the desired property is used in many implementations of SAT–solvers. Typically the heuristic consists of a more or less sophisticated strategy of selecting $x$ from $F$ such that the number of clauses in $F_x$ diminishes as far as possible compared to $F$ under a weighted measure, ranking short clauses which are deleted from $F$ higher than long clauses, see [1, 2].

In [3] it is shown that by the unit clause rule and a maximum occuring literal

rule alone (without backtracking in the search tree) a satisfying truth assignement will be found with probability tending to one for formulas in the class $cl(n,3)^r$, if $r < 2.9n$.

In section 2 we investigate the question what happens to solutions of CNF formulas $F$, if $F$ is split into $F_v$ and $F_{\overline{v}}$. The classes $cl(n,s)^r$ of formulas that we have in mind here mainly contain satisfiable formulas. Under the assumption of a uniform instance distribution we can show that the expected number of satisfying truth assignments of one of the two subformulas $F_v$ and $F_{\overline{v}}$ of $F \in cl(n,s)^r$ is significantly higher than that of the other subformula, see theorem 1. We show, e. g., for the classes of $cl(n,3)^{4n}$ that the expected number of satisfying truth assignments in the shorter of the two subformulas $F_v$ and $F_{\overline{v}}$ of $F \in cl(n,3)^{4n}$, is at least $\frac{11}{5}$ times higher than the corresponding number for the longer subformula. These results give additional theoretical support to the commonly used selection heuristics.

The results of section 2 suggest that solutions are distributed imbalanced among the subformulas into which $F$ is split during the computation of DPP and that there are more and less promizing subformulas. This phenomenon has been exploited in a mathematical model for explaining the superlinear speedup behaviour of parallel algorithms for solving the satisfiability problem, when applied to classes of formulas mainly containing satisfiable instances, [10]. This imbalance suggests a sequential DPP–based algorithm for the SAT–problem simulating a parallel SAT–solver with $k$ processors, using $k$ threads. Every thread performs the steps of a sequential DPP–based SAT–solver, described in [1]. In some fixed order the $k$ threads are activated. Every activated thread visits a fixed number $l = 500$ of leaves, until the next thread is activated. If a thread runs out of work in the search tree, it is restarted at a node on a smallest possible level that has not yet been visited. This $k$–thread–satisfiability solver has been implemented in $C$ and tested on a SUN Ultrasparc with a 167 MHz Ultrasparc CPU. The experimental results are reported in section 3.

# 2 The Imbalance of Distributions of Solutions of CNF Formulas

We will show that splitting formulas $F \in cl(n,s)^r$ into two subformulas by assigning both truth values to some Boolean variable $v \in V$, and evaluating $F$ according to these assignments yields two subformulas with significantly differing numbers of solutions in the average.

First we will introduce some notations which are convenient for our consideration. A formula $F \in cl(n,s)^r$ will be represented by an ordered multiset of clauses $c \in cl(n,s)$, where $cl(n,s) = \{x_{i_1} \vee \ldots \vee x_{i_s} : 1 \leq i_1 < \ldots < i_s \leq n, x_{i_j} \in \{v_{i_j}, \overline{v}_{i_j}\}\}$ denotes the set of clauses of length $s$ over the set of literals $L = \{v_1, \overline{v}_1, \ldots, v_n, \overline{v}_n\}$.

The formulas $F \in cl(n,s)^r$ are considered to be distributed uniformly. A truth assignment (t.a.) is a mapping $t : V \to L$, satisfying $t(v_i) \in \{v_i, \overline{v}_i\}$, and we identify $t$ and $t(V)$. $t(v_i) = v_i(\overline{v}_i)$ means that $v_i$ is assigned by true (false) under $t$. By $\overline{t} = L - t$ we denote the complementary t.a. of $t$. A t.a. $t$ is a solution of $F$, if for all $c \in F$, $t \cap c \neq \emptyset$ holds.

Set $v = v_n$. For $x \in \{v, \overline{v}\}$ and $F \in cl(n,s)^r, F_x := \{c - \{\overline{x}\} : c \in F, x \notin c\}$ denotes the subformula resulting from $F$ by assigning $t(v) = x$ (As usual, we have $\overline{\overline{v}} = v$.) Note that a solution $t$ of $F$ satisfies $|t| = n$, and a solution $t_1$ of $F_x$ satisfies $|t_1| = n-1$. By $\lambda(F), \lambda(F_x)$, we denote the number of solutions of $F, F_x$, resp. For $\mu \in \{min, max\}$, we define $\lambda_\mu(F) = \mu\{\lambda(F_v), \lambda(F_{\overline{v}})\}$. Obviously, the equation $\lambda(F) = \lambda_{empty}(F) = \lambda_{min}(F) + \lambda_{max}(F)$ holds. For $\mu \in \{min, max, empty\}$,

$$E(\lambda_\mu(n,s,r)) := |cl(n,s)|^{-r} \sum_{F \in cl(n,s)^r} \lambda_\mu(F)$$

denotes the expectation of the random variable $\lambda_\mu$ in the uniformly distributed class $cl(n,s)^r$.

**Proposition 1** *[5]* $\quad E(\lambda(n,s,r)) = 2^n(1-2^{-s})^r.$

□

Obviously the following holds:

$$E(\lambda(n,s,r)) = E(\lambda_{min}(n,s,r)) + E(\lambda_{max}(n,s,r)),$$

and

$$E(\lambda_{min}(n,s,r) \leq E(\lambda_{max}(n,s,r)).$$

We are looking for the ratio of

$$\rho(n,s,r) := \frac{E(\lambda_{max}(n,s,r))}{E(\lambda_{min}(n,s,r))}.$$

By definition, $\rho(n,s,r) \geq 1$, and the greater the value of $\rho(n,s,r)$ will be, the more unbalanced the solutions of formulas $F \in cl(n,s)^r$ will be distributed in the average among the subformulas $F_v$ and $F_{\overline{v}}$ of $F$. We will derive a (nontrivial) lower bound for $\rho(n,s,r)$.

When thinking about this problem, one may first conjecture that for growing values of $\lambda(n,s,r)$, $\rho(n,s,r) < 1 + \epsilon$ asymptotically holds for every $\epsilon > 0$. More precisely, one may expect that $\lambda_{max}(n,s,r) = \frac{1}{2}\lambda(n,s,r) + O(\sqrt{\lambda(n,s,r)})$.

This (wrong) argument is as follows. A solution $t$ of $F$ uniquely corresponds to a solution of $F_v$ or $F_{\overline{v}}$, depending on the value of $t(v)$. Furthermore a randomly chosen solution of an arbitrary formula $F \in cl(n,s)^r$ satisfies $Prob(t(v) = v) = Prob(t(v) = \overline{v}) = \frac{1}{2}$. Thus assuming all formulas $F \in cl(n,s)^r$ to have exactly $\lambda(n,s,r)$ solutions, each one corresponding to a solution of $F_v, F_{\overline{v}}$,

resp., which independently holds with probability $\frac{1}{2}$, implies that the average number of solutions of $F_v$ is $\frac{1}{2}\lambda(n,s,r)$ with a standard deviation of $\frac{1}{2}\sqrt{\lambda(n,s,r)}$. $F_v$ however, does not belong to $cl(n,s)^r$, so the expected number of solutions for $F_v$ is not $\frac{1}{2}\sqrt{\lambda(n,s,r)}$.

We now disprove this incorrect argument. Denote by $short(F)$ ($long(F)$) the shorter (longer) of the two subformulas $F_v$ and $F_{\overline{v}}$, for $F \in cl(n,s)^r$, and in case where $|F_v| = |F_{\overline{v}}|$, we fix $short(F) = F_v$ and $long(F) = F_{\overline{v}}$. Then

$$E(\lambda_{short}(n,s,r)) = |cl(n,s)|^{-r} \sum_{F \in cl(n,s)^r} \lambda(short(F))$$

denotes the expected number of solutions in the shorter subformula of $F$, for $F \in cl(n,s)^r$. $E(\lambda_{long}(n,s,r))$ is defined the same way. Obviously $E(\lambda_{min}(n,s,r)) \leq E(\lambda_{long}(n,s,r))$ and $E(\lambda_{max}(n,s,r)) \geq E(\lambda_{short}(n,s,r))$, therefore we have the following lower bound on $\rho(n,s,r)$:

$$\rho(n,s,r) \geq \varphi(n,s,r),$$

where

$$\varphi(n,s,r) := \frac{E(\lambda_{short}(n,s,r))}{E(\lambda_{long}(n,s,r))}.$$

**Theorem 1** *For $w \in \{short, long\}$ let $m_w = min$ $(max)$, if $w = short$ $(long)$. Then*

$$E(\lambda_w(n,s,r)) = 2^{n-1} \sum_{i=0}^{r} \binom{r}{i} (\frac{s}{n})^i (1-\frac{s}{n})^{r-i} 2^{-i} (1-2^{-s})^{r-i} \sum_{j=0}^{i} \binom{i}{j} (1-2^{1-s})^{m_w\{j,i-j\}}.$$

**Proof:** For $0 \leq j \leq i \leq r$ we define $K(i,j) := \{F \in cl(n,s)^r : \overline{v}$ occurs in exactly $j$ clauses and $v$ in excactly $i-j$ clauses of $F\}$. By definition the subsets $K(i,j), 1 \leq j \leq i \leq r$, partition the formula set $cl(n,s)^r$. For all $F \in K(i,j)$, we have $F_v \in cl(n-1,s)^{r-i} \times cl(n-1,s-1)^j$ and $F_{\overline{v}} \in cl(n-1,s)^{r-i} \times cl(n-1,s-1)^{i-j}$.

Conversely, for any two formulas $F_1, F_2 \in cl(n-1,s)^{r-i} \times cl(n-1,s-1)^j$ the following holds: if $F_1 \neq F_2$, then the sets $C_\vartheta = \{F \in cl(n,s)^r : F_v = F_\vartheta\}$, $\vartheta = 1,2$, satisfy $C_1 \cap C_2 = \emptyset$, $C_1, C_2 \subseteq K(i,j)$, and $|C_1| = |C_2|$. This means that by the operator $F \to F_v$ the uniform distribution defined on $K(i,j)$ is transferred into a uniform distribution on $cl(n-1,s)^{r-i} \times cl(n-1,s-1)^j$. Thus the average number of solutions of the formulas from the multiset $\{F_v : F \in K(i,j)\}$ is equal to the average number of solutions of the formulas from $cl(n-1,s)^{r-i} \times cl(n-1,s-1)^j$, which is $2^{n-1}(1-2^{-s})^{r-i}(1-2^{1-s})^j$, because of proposition 1. In the same way it can be seen that the formulas from the multiset $\{F_{\overline{v}} : F \in K(i,j)\}$ have $2^{n-1}(1-2^{-s})^{r-i}(1-2^{1-s})^{i-j}$ solutions in the average. If we restrict the functions "*short*" and "*long*" to formulas from $K(i,j)$, we obtain

$$short(K(i,j)) = \begin{cases} \{F_v & : F \in K(i,j)\}, \quad \text{if } j \leq i-j \\ \{F_{\overline{v}} & : F \in K(i,j)\}, \quad \text{else,} \end{cases}$$

and

$$long(K(i,j)) = \begin{cases} \{F_{\overline{v}} \ : F \in K(i,j)\}, & \text{if } j \leq i-j \\ \{F_v \ : F \in K(i,j)\}, & \text{else.} \end{cases}$$

Therefore the average number of solutions of the shorter (longer) subformula of the formulas in $K(i,j)$, denoted by $E(\lambda_{short}|K(i,j))$ $(E(\lambda_{long}|K(i,j)))$, are given by

$$E(\lambda_{short}|K(i,j)) = 2^{n-1}(1-2^{-s})^{r-i}(1-2^{1-s})^{min\{j,i-j\}}$$

and

$$E(\lambda_{long}|K(i,j)) = 2^{n-1}(1-2^{-s})^{r-i}(1-2^{1-s})^{max\{j,i-j\}}.$$

Since $v$ $(\overline{v})$ must occur in exactly $i-j$ $(j)$of the $r$ clauses of $F$ and the probability that a clause $c \in F$ contains $v$ or $\overline{v}$ is $\frac{s}{n}$, we get the following probability that a randomly chosen formula $F \in cl(n,s)^r$ belongs to $K(i,j)$:

$$Prob(F \in K(i,j)) = \binom{r}{i}(\frac{s}{n})^i(1-\frac{s}{n})^{r-i}\binom{i}{j}2^{-i}.$$

Thus we obtain

$$E(\lambda_{short}(n,s,r)) = \sum_{i=0}^{r}\sum_{j=0}^{i} E(\lambda_{short}|K(i,j))Prob(F \in K(i,j)).$$

$E(\lambda_{long}(n,s,r))$ is calculated the same way. □ Next we derive asymptotic lower bounds on $\varphi(n,s,r)$, for classes of formulas satisfying $r = cn, c > 0$

## An Asymptotic Lower Bound for $\varphi(n,s,cn)$

Let $s, c$ be constant and $n \geq 4s$. By theorem 1 and proposition 1 the following estimation holds

$$\begin{aligned} &\frac{E[\lambda_{short}(n,s,cn)]}{E[\lambda(n,s,cn)]} \\ &\geq \frac{1}{2}\sum_{i=0}^{4sc}\binom{cn}{i}\left(\frac{s}{n}\right)^i\left(1-\frac{s}{n}\right)^{cn}2^{-i}(1-2^{-s})^{-i}\sum_{j=0}^{i}\binom{i}{j}(1-2^{1-s})^{min(j,i-j)} \\ &\stackrel{*}{\to} \frac{1}{2}\sum_{i=0}^{4sc}\frac{(sc)^i}{i!}e^{-sc}2^{-i}(1-2^{-s})^{-i}\sum_{j=0}^{i}\binom{i}{j}(1-2^{1-s})^{min(j,i-j)} \\ &=: r(s,c). \end{aligned}$$

In $(*)$ we have used the limit $(1-\frac{s}{n})^{cn} \to e^{-sc}$. Now we express a lower bound for $\varphi(n,s,cn)$ in terms of $r(s,c)$. We have

$$\varphi(n,s,cn) \geq \frac{E[\lambda_{short}(n,s,cn)]}{E[\lambda_{long}(n,s,cn)]}$$

$$
\begin{aligned}
&= \frac{E[\lambda_{short}(n,s,cn)]}{E[\lambda(n,s,cn)] - E[\lambda_{short}(n,s,cn)]} \\
&= \frac{1}{\frac{E[\lambda(n,s,cn)]}{E[\lambda_{short}(n,s,cn)]} - 1} \\
&\geq \frac{1}{\frac{1}{r(s,c)} - 1} \qquad \text{(for } n \to \infty\text{)}.
\end{aligned}
$$

Since $n$ is not part of $r(s,c)$ we can calculate asymptotic lower bounds of $\varphi(n,s,cn)$ for fixed $s$ and $c$, see table 1 for $s = 3$ and table 2 for $s = 4$ (we have done this calculation with *Mathematica*).

| c | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\varphi(n,3,cn)$ | 1.460 | 1.734 | 1.977 | 2.208 | 2.434 |

Table 1: Lower bounds of the ratio $\varphi(n,3,cn)$.

| c | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|
| $\varphi(n,4,cn)$ | 1.684 | 1.757 | 1.829 | 1.890 | 1.967 | 2.035 |

Table 2: Lower bounds of the ratio $\varphi(n,4,cn)$.

# 3 Experimental Results

To demonstrate the effect of the imbalanced distribution of solutions on the runtime behaviour of a SAT-solver we applied our k-thread-satisfiability-solver described in section 1 to 500 randomly generated formulas from $cl(400,3)^{1600}$. Note that for $k = 1$ the algorithm is identical to the sequential SAT-solver described in [1]. The algorithm performs the Davis Putnam Procedure as described in section 1. The pure literal rule was not included in the algorithm since it slightly slowed down the runtime in our experiments. In the branching step a literal $x \in F$ is chosen according to the *lexicographic heuristic*, i.e. with lexicographic maximal vector

$$(H_1(x), H_2(x), \ldots, H_n(x))$$

where

$$H_i(x) := |\{c \in F \mid (x \in c \vee \overline{x} \in c) \wedge |c| = i\}|$$

The algorithm proceeds with $F_x$ first, iff $|F_x| \leq |F_{\overline{x}}|$. The implementation consists of highly optimized C code and uses efficient data structures. A single formula is kept in memory and modified in place to avoid copying. The modification of a formula $F$ into $F_x$ and vice versa (traversing an edge in the searchtree) is performed in time proportional to $size(F) - size(F_x)$ where $size(F) := \sum_{c \in F} |c|$.

In case of random formulas $F \in cl(n,s)^r$ $size(F) - size(F_x)$ is expected to be $rs(s+1)/(2n)$.

To exploit the imbalanced distribution of solutions in order to speed up the SAT-solver, our SAT-Solver generated a (predetermined) number of threads on a SUN with a 167MHz UltraSparc CPU. This way we were able to simulate a parallel computation with 1, 2, 4, 8, 16, 32 and 64 CPUs. The threads are scheduled round robin. After having visited 500 branch nodes a context switch is forced. The average runtimes and the average number of nodes that were expanded during the search are given in table 3 together with their standard deviation $\sigma$. From fig. 1 and 2 follows that the k-thread-satisfiability solver generates virtually no overhead, since the program always generates about 1800–2000 branch nodes per second.

| Threads | Time (sec) | $\sigma$(Time) | Nodes | $\sigma$(Nodes) |
|---|---|---|---|---|
| 1 | 370 | 1352 | 750169 | 2705539 |
| 2 | 304 | 1235 | 612274 | 2467132 |
| 4 | 225 | 986 | 441116 | 1929252 |
| 8 | 177 | 782 | 352990 | 1554555 |
| 16 | 154 | 752 | 291392 | 1446803 |
| 32 | 144 | 874 | 263199 | 1450585 |
| 64 | 182 | 1354 | 326946 | 2169765 |

Table 3: Runtimes and Branch Nodes

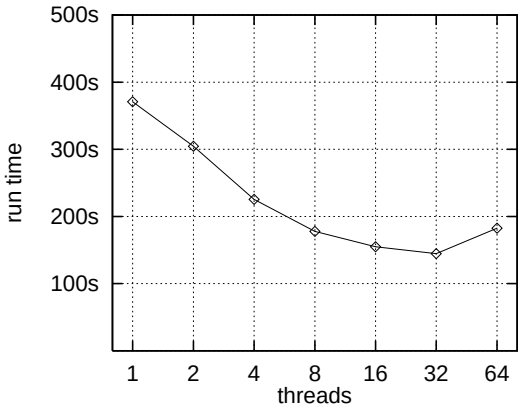


Figure 1: Runtime

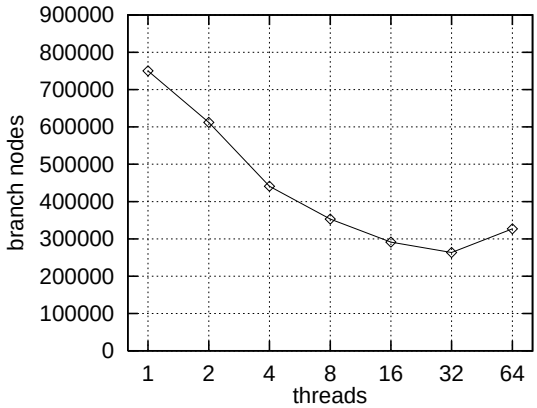


Figure 2: Branch nodes

# References


[1] M. Böhm and E. Speckenmeyer. A fast parallel SAT–solver — Efficient workload balancing. Technical Report 94-159, Angewandte Mathematik und Informatik, Universität zu Köln, 1994. Accepted for publication in: Annals of Mathematics and Artificial Intelligence.

[2] M. Buro and H. Kleine Büning. Report on a SAT competition. In *EATCS Bulletin*, volume 49, pages 143–151, February 1993.

[3] M. Chao and J. Franco. Probabilistic analysis of two heuristics for the 3-Satisfiability Problem. *SIAM Journal of Computation*, 15:1106–1118, 1986.

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

[5] J. Franco and M. Paull. Probabilistic analysis of the Davis–Putnam Procedure for solving the Satisfiability Problem. *Discrete Applied Mathematics*, 5:77–87, 1983.

[6] J. Gu. Efficient Local Search for Very Large–Scale Satisfiability Problems. *SIGART Bulletin*, 3:8–12, 1992.

[7] H. Kleine-Büning and T. H. Lettmann. *Aussagenlogik: Deduktion und Algorithmen.* B. G. Teubner, 1994.

[8] J. S. Schlipf, F. S. Annexstein, J. V. Franco, and R. P. Swaminathan. On finding solutions for extended Horn formulas. *Information Processing Letters*, 54:133–137, 1995.

[9] B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proc. of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, July 1992.

[10] E. Speckenmeyer, B. Monien, and O. Vornberger. Superlinear Speedup for Parallel Backtracking. In *Proc. Supercomputing 1987 (ICS '87)*, pages 985–993. Springer Verlag (LNCS 292), 1987.