# PURE LITERAL LOOK AHEAD: AN $O(1,497^n)$ 3-SATISFIABILITY ALGORITHM

(extended abstract)

Ingo Schiermeyer
Lehrstuhl für Diskrete Mathematik
und Grundlagen der Informatik
Technische Universität Cottbus
D-03013 Cottbus, Germany
E-mail: schierme@math.tu-cottbus.de

February 27, 1996

**Abstract**

In this paper we describe and analyse an improved algorithm for solving the 3-Satisfiability problem. The algorithm makes use of the concept 'Pure literal look ahead'. If $F$ is a boolean formula in conjunctive normal form with $n$ variables, then we will show that this algorithm solves the 3-Satisfiability problem in time less than $O(1,497^n)$.

## 1 INTRODUCTION

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of boolean variables. For each variable $v_i$ there is a *positive literal*, denoted by $v_i$, and a *negative literal*, denoted by $\overline{v}_i$. The literal $v_i$ has value *true* if and only if the variable $v_i$ has value *true*, and the literal $\overline{v}_i$ has value *true* if and only if the variable $v_i$ has value *false*. The literals $v_i$ and $\overline{v}_i$ will be called *complemented* and $L = \{v_1, \overline{v}_1, v_2, \overline{v}_2, \ldots, v_n, \overline{v}_n\}$ is the set of literals corresponding to $V$. For $L' \subseteq L$ we set $Lit(L') := \{v, \overline{v} \mid v \in L' \text{ or } \overline{v} \in L'\}$. A *k-clause* $(k \geq 1)$ is a subset of $k$ different literals of $L$. For a literal $v \in L$ let $n_k(v)$ denote the total number of occurrences of $v$ in the k-clauses of a formula $F$. A *truth assignment* (short: *assignment*) is a mapping $t$ which assigns each variable of $V$ the value *true* or the value *false*. For each variable $v$ we define: $t(v)$ has value *true* if and only if $t(\overline{v})$ has value *false*. An assignment $t$ *satisfies* a clause $K$ if and only if $t(x) = true$ for at least one literal $x$ of $K$; an assignment $t$ *satisfies* a *formula* $F$ of clauses if and only if it satisfies each clause of $F$. A formula $F$ is called *satisfiable* if there is at least one satisfying truth assignment for $F$.

For some $k$ with $1 \leq k \leq n$ we consider formulas $F$ with $r$ clauses $K_1, K_2, ..., K_r$ of lengths $k_1, k_2, ..., k_r$ in conjunktive normal form (CNF). The *Satisfiability Problem (Sat)* is the decision problem whether $F$ is satisfiable. If all clauses have length $k$ for some $k \geq 1$ then we speak of the *k-Satisfiability problem (k-Sat)*. For the purpose of this paper we assume that no clause contains complemented literals and that all clauses are pairwise different.

It is well-known that k-Sat is solvable in polynomial time for $k = 1, 2$ and $NP$-complete for $k \geq 3$ (cf. [Co],[GJ]). In [MS] and later in [S] two algorithms are presented to solve the 3-Satisfiability problem in less than $O(1,619^n)$ steps and in less than $O(1,579^n)$ steps, respectively. Recently, Kullmann [K] established a method with a very complex analysis (53 pages) to solve 3-Satisfiability in less than $O(1,5045^n)$ steps.

In this paper we present a simplified version of our algorithm given in [S]. The algorithm makes use of the concept 'Pure literal look ahead'. We introduce a new method to count the number of 1-clauses and 2-clauses. A precise calculation of the number of 1-clauses and 2-clauses then leads to the improvement of the constant to 1,497.

# 2 BASIC CONCEPTS

We call two formulas $F_1$ and $F_2$ *satisfiability equivalent* ($F_1 \equiv F_2$), if '$F_1$ is satisfiable $\Leftrightarrow$ $F_2$ is satisfiable'.

1. **Inclusion**
   If there are a $k_1 - clause$ $K_1$ and a $k_2 - clause$ $K_2$ such that $K_1 \subset K_2$ and $k_1 < k_2$, then every truth assignment satisfying $K_1$ also satisfies $K_2$ (but not vice versa). Therefore, $F \equiv F - K_2$ and we can delete $K_2$ from $F$. Our 3-Sat algorithm tests for inclusion and deletes all possible clauses.

2. **Independence**
   Two clauses $K_1$ and $K_2$ are called independent (cf. [D]), if every truth assignment, which does not satisfy $K_1$, satisfies $K_2$, and vice versa. If (e.g.) $K_1 = x_1 \vee x_2, K_2 = x_1 \vee \overline{x}_2 \vee x_3$ and $K_3 = x_1 \vee x_3$, then $K_1$ and $K_2$ are independent, whereas $K_1$ and $K_3$ are dependent. Now $K_1 \wedge K_2 \equiv K_1 \wedge K_3$ and thus $F \equiv F - K_2 + K_3$. In order to increase the number of 2-clauses our 3-Sat algorithm tests for independent pairs of a 2-clause and a 3-clause (like $K_1$ and $K_2$) and makes them dependent.

3. **Resolution**
For two clauses $A$ and $B$ and a variable $v$ such that $v \in A, \overline{v} \in B$ and $A$ and $B$ have no other pair of complemented literals, $(A - \{v\}) \vee (B - \{\overline{v}\})$ is called a resolvent of $A$ and $B$. It is well-known that $F \equiv F \wedge ((A - \{v\}) \vee (B - \{\overline{v}\}))$. If $A - \{v\} \equiv B - \{\overline{v}\}$ then $A$ and $B$ will be deleted from $F$ (inclusion). Our 3-Sat algorithm tests for resolution and adds all resolvents with at most three literals to $F$.

4. **Satisfiability equivalences**
Our 3-Sat algorithm makes use of the following satisfiability equivalences E1 - E5 which are introduced in [K] and can be easily verified. We may assume that there are no (more) 1-clauses.
**E1** Let $x$ be a variable such that $n_2(\overline{x}) = 1, n_3(\overline{x}) = 0$. Then for a clause $K = \overline{x} \vee a$
$$F \equiv F \wedge (x \vee \overline{a}).$$
**E2** Let $x$ be a variable such that $n_2(\overline{x}) = 0, n_3(\overline{x}) = 1$. Then for a clause $K = \overline{x} \vee a \vee b$
$$F \equiv F \wedge (x \vee \overline{a}) \wedge (x \vee \overline{b}).$$
**E3** Let $x$ be a variable such that $n_2(\overline{x}) = 0, n_3(\overline{x}) = 2$. Then for two clauses $K_1 = \overline{x} \vee a \vee b$ and $K_2 = \overline{x} \vee a \vee c$
$$F \equiv F \wedge (x \vee \overline{a}).$$
**E4** Let $x$ be a variable such that $n_2(\overline{x}) = 0, n_3(\overline{x}) = 2$. Then for two clauses $K_1 = \overline{x} \vee a \vee b$ and $K_2 = \overline{x} \vee \overline{a} \vee c$
$$F \equiv F \wedge (x \vee \overline{a} \vee \overline{c}) \wedge (x \vee \overline{b} \vee a) \wedge (x \vee \overline{b} \vee \overline{c}).$$
**E5** Let $x$ be a variable such that $n_2(\overline{x}) = 0, n_3(\overline{x}) = 2$. Then for two clauses $K_1 = \overline{x} \vee a \vee b$ and $K_2 = \overline{x} \vee c \vee d$
$$F \equiv F \wedge (x \vee \overline{a} \vee \overline{c}) \wedge (x \vee \overline{a} \vee \overline{d}) \wedge (x \vee \overline{b} \vee \overline{c}) \wedge (x \vee \overline{b} \vee \overline{d}).$$

The following 'special case S1' will be of major importance for our 3-Sat algorithm.

**S1** Suppose there is a 2-clause $K = x_1 \vee x_2$ such that $n(\overline{x}_1)$ and $n(\overline{x}_2)$ satisfy E5 and $n_3(x_1) = n_3(x_2) = 4$. Thus we may assume that $K_1 = \overline{x}_1 \vee a \vee b, K_2 = \overline{x}_1 \vee c \vee d \in F$. By resolution we conclude that $K_3 = \overline{x}_2 \vee \overline{a} \vee \overline{c}, K_4 = \overline{x}_2 \vee \overline{b} \vee \overline{d} \in F$ or $K_3 = \overline{x}_2 \vee \overline{a} \vee \overline{d}, K_4 = \overline{x}_2 \vee \overline{b} \vee \overline{c} \in F$. By symmetry it suffices to consider the first possibility. Then for $x_1 = x_2 = true$ we obtain the (new) 2-clauses $a \vee b, c \vee d, \overline{a} \vee \overline{c}, \overline{b} \vee \overline{d}$ with the only two possible truth assignments $a = d = true, b = c = false$ and $a = d = false, b = c = true$.

5. **Pure literal look ahead**
   For a literal $x$ let $N(x) := \{K - x \mid K \in F, x \in K\}$. Let $K$ be a k-clause, $k \geq 2$, such that $x_1 \vee x_2 \subseteq K$. If $N(x_1) \subseteq N(\overline{x}_2)$ in $F - K$, then assigning $x_2 = true$ gives the pure literal $\overline{x}_1$ in the reduced formula. This 'effect'has been applied successfully in the 3-Sat algorithm in [S] and will be called 'pure literal look ahead'.

6. **Autarkness**
   In [MS] Monien and Speckenmeyer introduced the fundamental concept of autarkness. Let $L' \subseteq L$ and $t : Lit(L') \to \{true, false\}$ be an assignment. Then $t$ is called *autark* in $F$ if and only if

   $$\forall K \in F : K \cap Lit(L') \neq \emptyset \Rightarrow \exists x \in K \cap Lit(L') : t(x) = true.$$

   The importance for their approach is based upon the following consideration. If $t : Lit(L') \to \{true, false\}$ is autark in $F$, then all clauses $K \in F$ with $K \cap Lit(L') \neq \emptyset$ are simultaneously satisfied by $t$ without having fixed any of the literals not in $Lit(L')$. If $t$ is not autark in $F$, then there is a clause $K \in F$ such that $K \cap Lit(L') \neq \emptyset$ and $\forall x \in K \cap Lit(L') : t(x) = false$. Thus the formula $F'$, which is computed by evaluating $F$ according to $t$ contains the clause $K' = K - Lit(L')$ with $k' < k$.

# 3 THE NEW 3-SAT ALGORITHM

Let $K_3(n, r)$ denote the set of all formulas $F$ which contain at most $n$ variables and $r$ clauses for which each clause $K \in F$ has at most three literals.

**3-Sat Algorithm**

Input: $F \in K_3(n, r)$
Output: $satisfiable$, if $F$ is satisfiable
$unsatisfiable$, else

Prozedur $Sat(F)$;

<u>begin</u>

1. <u>if</u> $F = \emptyset$ <u>then</u> $F$ is $satisfiable$;
   <u>if</u> $\square \in F$ <u>then</u> $F$ is $unsatisfiable$;

2. $M := \{y \in L \mid \overline{y} \notin L\}$; (* pure literals *)

if $M \neq \emptyset$ then
begin $F' := \{K \in F \mid K \cap M = \emptyset\}; Sat(F')$
if $F'$ satisfiable then $F$ is $satisfiable$
else $F$ is $unsatisfiable$
end

3. determine the length $l$ of a shortest clause in $F$;

4. if $l = 1$ then
   begin (4.)
   determine the set $M$ of all 1-clauses;
   if there exists some $i$, $1 \leq i \leq n$, such that $x_i, \overline{x}_i \in M$
   then $F$ is $unsatisfiable$
   else begin
   $F' := \{K - \overline{y} \mid K \in F, K \cap y = \emptyset\}$ for all $y \in M$; $Sat(F')$;
   if $F' satisfiable$ then $F$ is $satisfiable$
   else $F$ is $unsatisfiable$
   end;
   end (4.);

5. Test for 'Inclusion' and repeat;
   Test for 'Independence' and repeat;
   Test for 'Resolution' and repeat;
   Test for 'E1, E2' and repeat;
   Test for 'E3, E4, E5' and repeat;
   (* all tests are performed simultaneously *)

6. if $l = 2$ then
   begin (6.)
   if there are two clauses $K_1 = x_1 \vee x_2$ and $K_2 = \overline{x}_1 \vee \overline{x}_2$
   then begin (B1) (* $x_2 := \overline{x}_1$ *)
   $F' = \{(K \cup \overline{x}_1) - x_2 \mid K \in F, x_2 \in K, K \cap \overline{x}_2 = \emptyset, K \cap x_1 = \emptyset\};$
   $\cup\{(K \cup x_1) - \overline{x}_2 \mid K \in F, \overline{x}_2 \in K, K \cap x_2 = \emptyset, K \cap \overline{x}_1 = \emptyset\}$; $Sat(F')$;
   if $F'$ satisfiable then $F$ is $satisfiable$
   else $F$ is $unsatisfiable$
   end (B1);
   $branch := true$;
   while $branch$ do
   begin (B2-B4)
   $p := 2$;
   choose a variable $x_1$ such that $t := n_2(x_1) + n_2(\overline{x}_1) + 1$ is maximal

and such that $\mid n_2(x_1) - n_2(\overline{x}_1) \mid$ is minimal;
let $K_1 = x_1 \vee x_2, K_2 = x_1 \vee x_3, \ldots, K_{s-1} = x_1 \vee x_s$,
$K_s = \overline{x}_1 \vee x_{s+1}, \ldots, K_{t-1} = \overline{x}_1 \vee x_t$ be these $t-1$ clauses;
if $t = 2$ and $N(x_i) \subseteq N(\overline{x}_{3-i})$ for
$i = 1$ or $i = 2$ (* pure literal look ahead *)
<u>then</u> <u>begin</u> (B2)
  $T_1 := \{x_i, \overline{x}_{3-i}\}; T_2 := \{\overline{x}_i, x_{3-i}\}; branch := false;$
  <u>end</u> (B2);
if $t = 2$ and S1 holds with clauses $K_1 = \overline{x}_1 \vee a \vee b, K_2 = \overline{x}_1 \vee c \vee d$,
$K_3 = \overline{x}_2 \vee \overline{a} \vee \overline{c}, K_4 = \overline{x}_2 \vee \overline{b} \vee \overline{d}$
<u>then</u> <u>begin</u> (B3) (* $x_2 := \overline{x}_1$ *)
  $p := 3; T_1 := \emptyset; E_1 := \{x_2\}; D_1 := \{\overline{x}_1\};$
  $T_2 := \{x_1, x_2, a, \overline{b}, \overline{c}, d\}; T_3 := \{x_1, x_2, \overline{a}, b, c, \overline{d}\}; branch := false;$
  <u>end</u> (B3)
<u>else</u> <u>begin</u> (B4)
  $p := 2; T_1 := \{x_1, \overline{x}_{s+1}, \ldots, \overline{x}_t\};$
  $T_2 := \{\overline{x}_1, x_2, \ldots, x_s\}; branch := false;$
  <u>end</u> (B4);
<u>end</u> (B2-B4);

7. $naut := true; i := 1;$
<u>while</u> $i \leq p$ <u>and</u> naut <u>do</u>
  <u>begin</u>
  <u>if</u> $y = true$ for all $y \in T_i$ induces an autark
  truth assignment on $Lit(T_i)$ <u>then</u>
    <u>begin</u>
    $F_i := \{K \in F \mid K \cap Lit(T_i) = \emptyset\}; naut := false; Sat(F_i);$
    <u>if</u> $F_i$ satisfiable <u>then</u> $F$ is $satisfiable$;
        <u>else</u> $F$ is $unsatisfiable$;
    <u>end</u>
  <u>else</u> $i := i + 1;$
  <u>end</u>

8. <u>if</u> $naut$ <u>then</u>
  <u>begin</u>
  $unsat := true; i := 1;$
  <u>while</u> $i \leq p$ <u>and</u> $unsat$ <u>do</u>
    <u>begin</u>
    $F_i := \{K - \overline{T}_i \mid K \in F, K \cap T_i = \emptyset\}$; <u>if</u> $D_i \neq \emptyset$ <u>then</u> <u>begin</u>
    $F_i := \{(K \cup D_i) - E_i \mid K \in F_i, E_i \in K,$

$K \cap \overline{E}_i = \emptyset, K \cap \overline{D}_i = \emptyset\}$
$\cup\{(K \cup \overline{D}_i) - \overline{E}_i \mid K \in F_i, \overline{E}_i \in K,$
$K \cap E_i = \emptyset, K \cap D_i = \emptyset\}$;
$Sat(F_i)$;
<u>end</u>;
<u>if</u> $F_i$ satisfiable
<u>then</u> <u>begin</u>
$unsat := false$; $F$ is $satisfiable$;
<u>end</u>
<u>else</u> $i := i + 1$;
<u>end</u>
<u>if</u> $i = p + 1$ <u>then</u> $F$ is $unsatisfiable$;
<u>end</u>;
<u>end</u> (B1-B4)

9. <u>begin</u> (A2)
choose a clause $K = x_1 \vee x_2 \vee x_3$ such that $N(x_2) \subseteq N(\overline{x}_1)$ in $F - K$;
(* pure literal look ahead *)
if there are no such clauses then determine a clause $K = x_1 \vee x_2 \vee x_3$
such that $n(\overline{x}_1)$ is maximal; $naut := true; i := 1$;
<u>while</u> $i \leq 3$ <u>and</u> $naut$ <u>do</u>
<u>begin</u>
<u>if</u> $x_1 = \ldots x_{i-1} = false, x_i = true$ induces an
autark truth assignment on $Lit(\{x_1, \ldots, x_i\})$ <u>then</u>
<u>begin</u>
$F' := \{K \in F \mid K \cap Lit(\{x_1, \ldots, x_i\}) = \emptyset\}$;
$naut := false$; $Sat(F')$;
<u>if</u> $F'$ satisfiable <u>then</u> $F$ is $satisfiable$;
<u>else</u> $F$ is $unsatisfiable$;
<u>end</u>
<u>else</u> $i := i + 1$;
<u>end</u>

10. <u>if</u> $naut$ <u>then</u>
<u>begin</u> (10.)
$unsat := true; i := 1$;
<u>while</u> $i \leq 3$ <u>and</u> $unsat$ <u>do</u>
<u>begin</u>
$F_i := \{K - \{x_1, \ldots, x_{i-1}, \overline{x}_i\} \mid K \in F,$
$K \cap \{\overline{x}_1, \ldots, \overline{x}_{i-1}, x_i\} = \emptyset\}$; $Sat(F_i)$;

```
            if Fi satisfiable then
               begin
               unsat := false; F is satisfiable;
               end
            else i := i + 1;
            end;
      if i = 4 then F is unsatisfiable
      end (10.)
   end (A2)

end;
```

# 4 COMPLEXITY-ANALYSIS

**Lemma 4.1** *Let $F \in K_3(n,r)$. Then for all subproblems $F_i$ generated in the while-loop of statement 8. or 10. there is a clause $K_j \in F_i$ with $k_j < k$.*

**Proof:** If statement 8. will be executed then the following must have happened during the preceeding execution of statement 7.. For each possible branching the assignments $t_i$ are not autark, which are given by $x_j = true$ for all $x_j$ of $T_i$. Hence there is a clause $K \in F$ such that $K \cap Lit(T_i) \neq \emptyset$ and $\forall x \in (K \cap Lit(T_i)) : t_i = false$. Thus $K \cap \overline{T_i} = \emptyset$ and therefore $K_j := (K - T_i) \in F_i$, where $k_j < k$.

If 10. will be executed then during the execution of statement 9. for all $i$ with $1 \leq i \leq l$ the assignment $t_i : x_1 = \ldots = x_{i-1} = false, x_i = true$ was not autark in $F$. Hence there is a clause $K \in F$, such that $K \cap Lit(\{x_1, \ldots, x_i\}) \neq \emptyset$ and $\forall x \in K \cap Lit(\{x_1, \ldots, x_i\} : t_i(x) = false$. Thus $K \cap \{\overline{x}_1, \ldots, \overline{x}_{i-1}, x_i\} = \emptyset$ and therefore $K_j := (K - \{x_1, \ldots, x_{i-1}, \overline{x}_i\} \in F_i$, where $k_j < k$.

□

**Lemma 4.2** *The execution time of Sat between two successive calls is bounded by a polynom $p(n)$ of $n$ when applied to $F \in K_3(n,r)$.*

**Proof:** The total number of 1-clauses, 2-clauses and 3-clauses is bounded by $O(n^3)$. Thus the total number of pairs of clauses is bounded by $O(n^6)$. Hence altogether the execution time can be bounded by a polynom $p(n)$ of $n$.

□

In order to compute the running time of the algorithm 3-Sat it remains to compute the number $T(F)$ of recursive calls $Sat$ has to perform to evaluate $F$. Define

$$\begin{aligned} T_3(n) &:= max\{T(F) \mid F \in K_3(n,r)\} \text{ and} \\ T_3'(n) &:= max\{T(F) \mid F \in K_3(n,r), \exists K_i \in F : k_i < 3\}. \end{aligned}$$

For a formula $F \in K_3(n,r)$ let $p$ be the maximal number of 1-clauses and 2-clauses (with preference of 1-clauses) such that no variable occurs more than twice (2-partition). Let $q$ be the number of remaining 2-clauses and define $m := p + min(2,q)$. In order to estimate $T_3(n)$ and $T_3'(n)$ we choose two functions $F_3(n)$ and $F_3'(n)$ as follows:

$$\begin{aligned} F_3(n) &= c\beta^n \cdot \frac{\beta}{\alpha^4} \\ F_3'(n) &= c\beta^n \alpha^{-m}, \end{aligned}$$

where $\beta = 1,4963, \alpha = 1,04855$ and $c$ is a sufficiently large constant.

**Lemma 4.3**

$$T_3(n) \leq F_3(n)$$

**Proof:** extract

If there is an autark truth assignment with $i$ variables then

$$T(F) \leq 1 + c\beta^{n-i} \cdot \frac{\beta}{\alpha^4} < c\beta^n \cdot \frac{\beta}{\alpha^4}.$$

So assume that there occur no autark truth assignments. If $N(x_2) \subseteq N(\overline{x}_1)$ in $F - K$, then $x_2 = false$ in $F_1$. Hence

$$\begin{aligned} T(F) &= 1 + T(F_1) + T(F_2) \\ &\leq 1 + (1 + c\beta^{n-2}\alpha^{-2}) + c(\beta^{n-2}\alpha^{-2} + \beta^{n-3}\alpha^{-2}) \\ &= 2 + c\beta^n \cdot \frac{1}{\alpha^2}(\frac{2}{\beta^2} + \frac{1}{\beta^3}) \\ &< c \cdot 1,0839 \cdot \beta^n < c\beta^n \cdot \frac{\beta}{\alpha^4}. \end{aligned}$$

□

**Lemma 4.4**

$$T_3'(n) \leq F_3'(n)$$

**Proof:** extract

If there is an autark truth assignment with $i$ variables, then by the definition of $m$ at most $2i+2$ 1-clauses and 2-clauses are deleted. Hence $i \geq \frac{m}{4}$ and thus

$$T(F) \leq 1 + c\beta^{n-i} = 1 + c\beta^n \cdot \beta^{-i} < c\beta^n\alpha^{-m}.$$

So assume that there occur no autark truth assignments.
Case $\mathbf{t} = \mathbf{s} = \mathbf{2}$
**3.5** If $n_3(x_1) = n_3(x_2) = 4$ then by S1 we obtain

$$\begin{aligned} T(F) &\leq 1 + c\beta^{n-1}\alpha^{-(m-1)} + 2 \cdot c\beta^{n-6}\alpha^{-(m-5+1)} \\ &= 1 + c\beta^n\alpha^{-m}(\frac{\alpha}{\beta} + \frac{2\alpha^4}{\beta^6}) < c \cdot 0,9162 \cdot \beta^n\alpha^{-m} \end{aligned}$$

□

With lemmata 4.2, 4.3 and 4.4 we obtain the main result of this chapter.

**Theorem 4.5** *The algorithm* $3 - Sat$ *computes every formula* $F \in K_3(n,r)$ *in* $O(1,4963^n \cdot p(n))$ *steps.*

**Corollary 4.6** *The algorithm* $3 - Sat$ *computes every formula* $F \in K_3(n,r)$ *in less than* $O(1,497^n)$ *steps.*

# References


[Co] S. A. Cook, *The Complexity of Theorem-Proving Procedures*, Proc. 3rd Ann. ACM Symp. on Theory of Computing, New York, 1971, 151 - 158.

[D] O. Dubois, *Counting the number of Solutions for Instances of Satisfiability*, Theor. Comput. Sci. 81 (1991) 49 - 64.

[GJ] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

[K] O. Kullmann, *Methods for 3-SAT-decision in less than* $1,5045^n$ *steps*, preprint 1994.

[MS] B. Monien and E. Speckenmeyer, *Solving Satisfiability in less than* $2^n$ *Steps*, Discrete Appl. Math. 10 (1985) 287 - 295.

[S] I. Schiermeyer, *Solving 3-Satisfiability in less than* $1,579^n$ *Steps*, Selected papers from CSL'92, LNCS Vol. 702 (1993) 379 - 394.