Solving SAT for CNF formulas with a one-sided restriction on variable occurrences

Daniel Johannsen[1], Igor Razgon[2], and Magnus Wahlström[1]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2] Cork Constraint Computation Centre, University College Cork, Ireland

Abstract. In this paper we consider the class of boolean formulas in Conjunctive Normal Form (CNF) where for each variable all but at most d occurrences are either positive or negative. This class is a generalization of the class of CNF formulas with at most d occurrences (positive and negative) of each variable which was studied in [Wahlström, 2005]. Applying complement search [Purdom, 1984], we show that for every d there exists a constant $\gamma_d < 2 - \frac{1}{2d+1}$ such that satisfiability of a CNF formula on n variables can be checked in runtime $O(\gamma_d^n)$ if all but at most d occurrences of each variable are either positive or negative. We thoroughly analyze the proposed branching strategy and determine the asymptotic growth constant γ_d more precisely. Finally, we show that the trivial $O(2^n)$ barrier of satisfiability checking can be broken even for a more general class of formulas, namely formulas where the positive or negative literals of every variable have what we will call a *d–covering*. To the best of our knowledge, for the considered classes of formulas there are no previous non-trivial upper bounds on the complexity of satisfiability checking.

1 Introduction

Design of fast exponential algorithms for satisfiability checking has attracted considerable attention of various research communities in applied as well as in theoretical fields of computer science. Since it is unknown how to break the trivial $O(2^n)$ barrier for the runtime of the unrestricted satisfiability problem (SAT) on n variables, current research concentrates on more efficient satisfiability checking of restricted classes of conjunctive normal form (CNF) formulas. The most widely considered restriction is k–SAT, including all formulas whose maximal clause length is at most k. Currently the best runtime is $O\big((2-\frac{2}{k+1})^n\big)$ for a deterministic algorithm [3], and a stronger but comparable bound for a randomized one [4]. However, to improve the understanding of how the structure of a CNF formula influences the efficiency of satisfiability checking, it is important to study further sub-classes of CNF formulas. Two such sub-classes are formulas with restrictions on (i) the value of the density m/n (where m is the number of clauses) and (ii) on the number of occurrences of each variable. Calabro, Impagliazzo, and Paturi [2] proved that for both classes satisfiability can be checked in runtime $O(\gamma^n)$ with $\gamma < 2$, and asymptotically related the

bounds of k–SAT, SAT with $m/n \leq \Delta$, and SAT with at most d occurrences per variable to one another, essentially showing that the two latter bounds behave as the former one with $k = \Theta(\log \Delta)$ and $k = \Theta(\log d)$, respectively. For small values of d, Wahlström [7, 6] has given a stronger bound for formulas with at most d occurrences per variable of order $O(1.1279^{(d-2)n})$.

The main contribution of this paper is extending the boundaries of the classes of CNF formulas for which satisfiability can be checked faster than in runtime $O(2^n)$. In particular, we continue the line of research of [7], where the total number of occurrences (positive *and* negative) of each variable in a CNF formula F was restricted to at most d. Now, for each variable x of F we call the literal $\ell \in \{x, \neg x\}$ with less occurrences in F *minor* and its negation *major*. We then only restrict the number of minor literals in F to be at most d, that is, the total number of literal occurrences per variable in F remains unrestricted.

We first study the satisfiability of formulas where for each variable all but one occurrences are positive or negative and give an algorithm with runtime $O^*(3^{n/3})$ (Theorem 1)[3]. Then, we propose an algorithm based on complement search [5] and show that for each fixed d there exists a constant $\gamma_d < 2$ such that the satisfiability of CNF formulas with at most d minor literals per variable can be checked in runtime $O(\gamma_d^n)$ (Theorem 2). Next, we investigate the main parameter of the algorithm closer and bound γ_d by $2 - \frac{1}{2d+1}$ (Theorem 3). Finally, we further generalize the class of CNF formulas with at most d minor literals per variable to a class we call *CNF formulas with a d–covering* and present an algorithm checking satisfiability of such formulas in runtime $O^*(\Phi_{d+1}^n)$ (Theorem 4), where $\Phi_d^n < 2$ is the Fibonacci constant of order d as defined below.

2 SAT for CNF formulas with unique minor literals

In this section we study CNF formulas with exactly one minor literal per variable.

Lemma 1. *A CNF formula with at most one minor literal per variable either contains a clause consisting only of minor literals or is trivially satisfied by satisfying all major literals.*

This basic observation allows us to check the satisfiability of such formulas.

Algorithm 1
Input: *CNF formula F with set of minor literals M, each occurring once.*
Output: TRUE *if F is satisfiable, otherwise* FALSE.
If *F is empty* **then** *return* TRUE.
If *F contains an empty clause* **then** *return* FALSE.
If *F contains a clause $C = (\ell_1 \vee \ldots \vee \ell_r)$ with $\ell_1, \ldots, \ell_r \in M$* **then**
  let $C_i = C[\ell_i = \text{FALSE}]$[4] *for* $i \in \{1, \ldots, r\}$ **and**
  try branches $F[\ell_i = \text{TRUE}, C_i = \text{FALSE}]$ *for* $i \in \{1, \ldots, r\}$
else *evaluate* $F[\ell = \text{FALSE}$ *for all* $\ell \in M]$.

[3] We use the $O^*(\cdot)$ notation to suppress factors polynomial in the length of the input.
[4] For a literal ℓ or a clause C we denote by $F[\ell \,/\, C = \text{TRUE} \,/\, \text{FALSE}]$ the residual formula obtained by assigning TRUE or FALSE to ℓ or all literals of C, respectively. If this assigns all literals of a clause to FALSE, then leave the clause as an empty clause (indicating that a contradiction has been encountered).

According to Lemma 1 this algorithm is correct. The runtime is dominated by the branching strategy and thus of order $\max_{r\in\mathbb{N}} r^{n/r}$ which is worst at $r = 3$.

Theorem 1. *Let $n \in \mathbb{N}$ and let F be a CNF formula on n variables with at most one minor literal per variable. Then Algorithm 1 checks the satisfiability of F in runtime $O^*(3^{n/3}) \subseteq O(1.4423^n)$.*

Correspondingly, we can express a pseudo-lower runtime bound of $\Omega(2^{n/2})$ which holds if the general SAT problem cannot be solved faster than in $\Omega(2^n)$.

Lemma 2. *Let $n \in \mathbb{N}$ and $\gamma > 1$. If the satisfiability of any CNF formula on n variables with at most one minor literal per variable can be checked in runtime $O(\gamma^n)$ then the satisfiability of any (unrestricted) CNF formula on n variables can be checked in runtime $O(\gamma^{2n})$.*

Proof. Every general CNF formula on n variables can be equivalently transformed to a CNF formula on $2n$ variables such that each minor literal occurs at most once. For each variable, replace all minor literals by a new variable and add a clause containing the minor literal and the negation of the new variable.

A similar reduction from 3-CSP improves this pseudo-lower bound to $\Omega(3^{n/3})$ but is omitted due to lack of space.

3 SAT of CNF formulas with at most *d* minor literals

The following lemma by Purdom [5] allows us to recursively check the satisfiability of CNF formulas with n variables and at most d minor literals per variable in runtime $O(\gamma^n)$ with $\gamma < 2$.

Lemma 3. *Let ℓ be a literal of a CNF formula F. Then either $F[\ell = \textsc{False}]$ is satisfiable or for all satisfying assignments of F there is a clause C containing ℓ such that all literals of C are assigned* False *except for ℓ which is assigned* True.

The previous lemma allows us to check the satisfiability of CNF formulas with at most d minor literals per variable by using a branching strategy parameterized by the *branching threshold* k for short clauses.

Algorithm 2
Parameter: *Branching threshold k for short clauses.*
Input: *CNF formula F with set of minor literals M*
Output: True *if F is satisfiable, otherwise* False.
If *F is empty* **then** *return* True.
If *F contains an empty clause* **then** *return* False.
If *F contains a clause $C = (\ell_1 \vee \ldots \vee \ell_r)$ with $r \leq k$* **then**
  try branches $F[\ell_1 = \ldots = \ell_{i-1} = \textsc{False}, \ell_i = \textsc{True}]$ *for* $i \in \{1, \ldots, r\}$
else
  pick $\ell \in M$ *contained in the clauses* $C_1, \ldots, C_s$ **and**
  let $C'_i = C_i[\ell = \textsc{False}]$ *for* $i \in \{1, \ldots, s\}$ **and**
  try branches $F[\ell = \textsc{True}]$ *and* $F[\ell = \textsc{True}, C'_i = \textsc{False}]$ *for* $i \in \{1, \ldots, s\}$.

Let $T(n)$ be the runtime of the algorithms branching procedure (where n is the number of variables). The runtime of the branch where the shortest clause is of size $r \leq k$ is of order $T(n-1) + \ldots + T(n-r)$ which is at most

$$T_A(n) = \sum_{i=1}^{k} T(n-i) \tag{1}$$

The growth constant of this recursion is known to be the k–th order Fibonacci number Φ_k (see, e. g., [8]). That is, $T_A(n) \in O^*(\Phi_k^n)$, where Φ_k is the unique solution of the equation $x^k(2-x) = 1$ in the interval $(1,2)$. The number Φ_2 is the golden ratio $(1+\sqrt{5})/2$, more initial values of Φ_k are given in Table 1.

The runtime of the branch where the shortest clause is of size at least $k+1$ is at most $T_B(n) = T(n-1) + s \cdot T(n-(k+1))$ where s is the number of clauses containing the eliminated literal. Thus, for $s \leq d$, this runtime is at most

$$T_B(n) = T(n-1) + d \cdot T(n-1-k)\,. \tag{2}$$

Hence, the maximum of $T_A(n)$ and $T_B(n)$ is an upper bound on the runtime $T(n)$ of Algorithm 2 with parameter k on CNF formulas with at most d minor literals per variable. Obviously, $T(n)$ is strongly influenced by the choice of k. For example, if we choose $k = d+1$, then $T_A(n)$ dominates $T_B(n)$.

Theorem 2. *Let $n, d \in \mathbb{N}$ and let F be a CNF formula on n variables with at most d minor literals per variable. Then Algorithm 2 with parameter $d+1$ checks the satisfiability of F in runtime $O^*(\Phi_{d+1}^n)$.*

In the remainder of this section we see, how to choose the parameter k optimally for every fixed $d \in \mathbb{N}$. Suppose that k is also fixed. Then $T(n)$ is of order γ^n, where $\gamma \in (1,2)$ is the smallest constant that satisfies both recursions (1) and (2).

Lemma 4. *Let $n, d, k \in \mathbb{N}$ with $d \geq 2$ and let F be a CNF formula on n variables with at most d minor literals per variable. Then Algorithm 2 with parameter k checks the satisfiability of F in runtime $O^*(\gamma^n)$ for all $\gamma \in (1,2)$ with*

$$\frac{d}{\gamma - 1} \leq \gamma^k \leq \frac{1}{2-\gamma}\,. \tag{3}$$

Proof. The statement follows by induction on n. □

A direct consequence of the previous lemma is that the minimal γ satisfying condition (3) dominates the growth constant of $T(n)$ for given d and k. Clearly, the condition is satisfied for every $d \geq 2$ and $k \in \mathbb{N}$ as γ tends to two. On the other hand, as γ tends to one, eventually one of the two inequalities is violated. Thus, a minimal γ satisfies at least one of the two equations with equality.

On the other hand, if there exists a k such that the corresponding γ satisfies both inequalities, then γ is optimal for all values of k (decreasing k violates the first inequality while increasing k violates the second one; in both cases we need to increase γ to satisfy condition (3) again). This situation occurs if $\gamma = 2 - \frac{1}{d+1}$. In this case the lower and the upper bound on γ^k both have the value $d+1$. For smaller values of γ, condition (3) can never be satisfied.

Lemma 5. *Let $d \in \mathbb{N}$ with $d \geq 2$. Then $2 - \frac{1}{d+1}$ is a lower bound on all γ satisfying condition (3) for any $k \in \mathbb{N}$.*

Note that this lower bound is not necessarily attained since k is integral. If we drop this condition, then for $k_d^* = \log(d+1)/(\log(2d+1) - \log(d+1))$ both inequalities in condition (3) become equalities. For $k \geq \lceil k_d^* \rceil$, the right inequality in condition (3) is violated for $\gamma = 2 - \frac{1}{d+1}$, while the left right inequality is satisfied. For $k \leq \lfloor k_d^* \rfloor$ the opposite holds. The further k is apart from k_d^*, the more we have to increase γ to satisfy the violated inequality. Thus one of $\lfloor k_d^* \rfloor$ and $\lceil k_d^* \rceil$ is optimal, depending for which the corresponding γ is smaller. Table 1 lists k_d, γ_A and γ_B for the initial values of d.

Lemma 6. *Let $d \geq 2$ and let $k_d^* = \frac{\log(d+1)}{\log(2d+1) - \log(d+1)}$. Moreover, let γ_A and γ_B be the unique solutions of $(2 - \gamma_A)\,\gamma_A^{\lceil k_d^* \rceil} = 1$ and $(\gamma_B - 1)\,\gamma_B^{\lfloor k_d^* \rfloor} = d$ in the interval $(1, 2)$. Then $\gamma_d = \min\{\gamma_A, \gamma_B\}$ satisfies condition (3) for $k_d \in \{\lceil k_d^* \rceil, \lfloor k_d^* \rfloor\}$ chosen respectively. Furthermore, in this γ_d is minimal for all k.*

Finally, we show a closed form upper bound for the runtime of Algorithm 2.

Theorem 3. *Let $n, d \in \mathbb{N}$ with $d \geq 2$ and k_d and γ_d as in Lemma 6. Then Algorithm 2 with parameter k_d checks the satisfiability of a CNF formula on n variables with at most d minor literals per variable in runtime $O\big((2 - \frac{1}{2d+1})^n\big)$.*

Proof. For $d \geq 2$, γ is smaller than $d/(\gamma - 1)$ and strictly smaller than $1/(2 - \gamma)$ divided by $d/(\gamma - 1)$. Hence, there exists a $k \in \mathbb{N}$ such that $\gamma = 2 - \frac{1}{2d+1}$ satisfies condition (3). The statement follows from Lemma 4 and Lemma 5.

4 Further generalization

We can further generalize Lemma 3 to obtain an algorithm of runtime $O(\gamma^n)$ with $\gamma < 2$ for a class of CNF formulas which neither have short clauses nor a one-sided restriction on the variable occurrences.

d	k_d	k_d^*	γ_A	γ_B	$2 - \frac{1}{d+1}$	$2 - \frac{1}{2d+1}$	Φ_{d+1}
2	2	2.15064	**1.69562**	1.83929	1.66667	1.80000	1.83929
3	3	2.47720	1.86371	**1.83929**	1.75000	1.85714	1.92756
4	3	2.73817	2.00000	**1.83929**	1.80000	1.88889	1.96595
5	3	2.95602	2.11634	**1.83929**	1.83333	1.90909	1.98358
6	3	3.14343	**1.88947**	1.92756	1.85714	1.92308	1.99196

Table 1. Runtime bounds for Algorithm 2 with parameter k on CNF formulas with at most d minor literals per variable. For $d = 2, \ldots, 6$ the table shows the following values: the optimal choice of the branching threshold k_d and the corresponding relaxed real-valued optimum k_d^*; the two choices of the optimal growth constant γ_A and γ_B (Lemma 6) with the better one in bold face; the lower bound $2-1/(d+1)$ (Lemma 5), the upper bound $2-1/(2d+1)$ (Theorem 3), and the weak upper bound Φ_{d+1} (Theorem 2).

A *covering* of a literal ℓ in a CNF formula F is a set L of literals, such that (i) no literal and its negation are both in L, (ii) ℓ is not in L, and (iii) all clauses of F containing ℓ also contain a literal of L. We say that F has a *d–covering* if one of the two literals corresponding to each variable has a covering of size d.

It is not hard to see that a CNF formula with clauses of size at least $d+1$ and at most d minor literals per variable has a d–covering. But, this weaker condition is still sufficient for breaking the $O(2^n)$ runtime barrier.[5]

Algorithm 3
Input: *CNF formula F with set of minor literals M*
Output: TRUE *if F is satisfiable, otherwise* FALSE.
If *F is empty* **then** *return* TRUE.
If *F contains an empty clause* **then** *return* FALSE.
Pick *literal ℓ_0 covered by $\ell_1, \ldots \ell_d$* **and**
try branches $F[\ell_0 = \ldots = \ell_{i-1} = \text{TRUE}, \ell_i = \text{FALSE}]$ *for* $i \in \{0, \ldots, d\}$

According to Lemma 3 and the definition of a d–covering of a CNF formula F, any satisfying assignment of F also satisfies the clause $(\neg\ell_0 \vee \neg\ell_1 \vee \cdots \vee \neg\ell_d)$.

Theorem 4. *Let $n, d \in \mathbb{N}$. Then Algorithm 3 checks the satisfiability of a CNF formula on n variables that has a d–covering in runtime $O^*(\Phi^n_{d+1})$.*

References

1. N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.
2. C. Calabro, R. Impagliazzo, and R. Paturi. A duality between clause width and clause density for SAT. *Computational Complexity, Annual IEEE Conference on*, 0:252–260, 2006.
3. E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. M. Kleinberg, C. H. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2 - 2/(k + 1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, 289(1):69–83, 2002.
4. R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005.
5. P. W. Purdom. Solving satisfiability with less searching. *IEEE Trans. Pattern Anal. Machine Intell.*, 6(4):510–513, 1984.
6. M. Wahlström. An algorithm for the SAT problem for formulae of linear length. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA-2005)*, pages 107–118, 2005.
7. M. Wahlström. Faster exact solving of SAT formulae with a low number of occurrences per variable. In *Proceedings of SAT*, pages 309–323, 2005.
8. D. Wolfram. Solving generalized Fibonacci recurrences. *The Fibonacci Quarterly*, 36.2:129–145, 1998.

[5] For fixed $d \in \mathbb{N}$ we can test for a d–covering of a CNF formula (and also find it) in runtime $O(n^{f(d)})$. However, Parameterized Complexity Theory suggests also a lower bound of $\Omega(n^d)$. Furthermore, the problem to compute a minimal covering of a given CNF formula is a generalization of the hitting set problem, which in polynomial time cannot be approximated better than within a factor $\Theta(\log n)$ unless P=NP [1].