Technical Reports on Mathematical and Computing Sciences: TR-C146

title: An Improved Randomized Algorithm for 3-SAT

author: Rainer Schuler[1], Uwe Schöning[1], and Osamu Watanabe[2]

affiliation:

1. Abt. Theoretische Informatik, Universitat Ulm
   Oberer Eselsberg, D 89069 Ulm, Germany
2. Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
   Meguro-ku Ookayama, Tokyo 152-8552

email: watanabe@is.titech.ac.jp

acknowledgments to financial supports:

The second author is supported in part by DFG grant Sch 302/5-2.
The third author is supported in part by JSPS/NSF cooperative research:
Complexity Theory, for Strategic Goals, 1998–2001.

Abstract. Schöning [Sch99] proposed a simple yet efficient randomized algorithm for solving the k-SAT problem. His analysis showed that for 3-SAT, finding a satisfying assignment of any satisfiable formula F with n variables can be achieved in $\mathrm{poly}(n) \cdot (4/3)^n$ ($= \mathrm{poly}(n) \cdot (1.3333)^n$) expected time, which is optimal up to now. In this paper, we improve this expected time bound by using a combination of a deterministic algorithm for 3-SAT and an improvement of the above randomized algorithm. Our new bound for the above task is $\mathrm{poly}(n) \cdot (1.3303)^n$.

1 Introduction and Preliminaries

We present a randomized algorithm for 3-SAT that improves the efficiency of the known fastest randomized algorithm of Schöning [Sch99].

The k-satisfiability problem (in short, k-SAT) is to determine whether a given Boolean formula F in the k-CNF is satisfiable. This problem has been studied by many researchers, and various algorithms have been proposed. Some algorithms indeed have a worst-case time complexity much better than the trivial exhaustive search algorithm; see, e.g., [MS85, PPSZ98, Sch99, Kul99, DGHS00, Hir00a, Hir00b]. In particular, the algorithm proposed by Schöning [Sch99], which is for the k-SAT problem in general, has so far the best worst-case time bound for 3-SAT. His analysis showed that for 3-SAT, finding a satisfying assignment of any satisfiable formula F with n variables can be achieved in $\mathrm{poly}(n) \cdot (4/3)^n$ ($= \mathrm{poly}(n) \cdot (1.3333)^n$) expected time, which is optimal up to now. (By $\mathrm{poly}(n)$, we mean some polynomial over n. We may assume that the number of clauses is bounded by $O(n^3)$; hence, we estimate the time bound in terms of n.) In this paper, we improve this expected time bound by using a combination of a new simple deterministic algorithm for 3-SAT and an improvement of the above randomized algorithm. Our new combined algorithm finds a satisfying assignment (if exists) in $\mathrm{poly}(n) \cdot (1.3303)^n$ expected time.

Here we review Schöning's randomized algorithm for k-SAT. First recall some basic definitions on k-SAT. A formula F is in the *k-conjunctive normal form* (in short, *k-CNF*) if F is a

```
Program Search(input F);    % F is a 3-CNF formula with n variables.
  repeat
    select an initial assignment a uniformly at random from {0,1}^n;
    for 3n times do
      if F(a) = 1 then accept (and halt);
      C ← a clause of F that is not satisfied by a;
      % Though choosing a shortest clause for C would be a good heuristic,
      % for our theoretical analysis, we may choose any unsatisfied clause.
      select one literal of C uniformly at random
        and flip the assignment to this literal;
    end-for;
  end-repeat;
end.
```

Figure 1: Schöning's randomized local search algorithm

conjunction of *clauses* which contain at most k *literals*, i.e., F is of the form $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where $C_i = l_{j_1} \vee l_{j_2} \vee \cdots \vee l_{j_k}$ and l_{j_h} is either a variable or its negation. Throughout this paper, we will use n and m to denote the number of variables and that of clauses of a given input formula. An *assignment* of a Boolean formula is a 0/1 assignment to every variable of the formula; in this paper, we will regard an assignment as an element of $\{0,1\}^n$, a binary sequence of length n. The problem k-SAT is to determine whether a given formula in the k-CNF has a *satisfying assignment*, an assignment that makes the formula true. Throughout this paper, we consider 3-SAT, and by "formula" we always mean a 3-CNF formula.

Schöning's algorithm is a randomized local search algorithm that is described as Figure 1. Note that the algorithm (as stated in the figure) is designed for searching a satisfying assignment, and it does not terminate until a satisfying assignment is found. Hence, it runs forever if an unsatisfiable formula is given as an input. Note, however, as shown in the theorem below the *success probability* of one repeat-iteration, i.e., the probability of finding a satisfying assingment (if it exists) during one repeat-iteration, is at least $1/T(n)$ for some $T(n)$. This means that if an input formula is satisfiable, then the probability that executing the repeat-iteration $N \cdot T(n)$ times and not finding a satisfying assingment is at most $(1 - 1/T(n))^{N \cdot T(n)}$, which is less than e^{-N}. Therefore, one can modify the algorithm so that it halts and rejects an input after executing the repeat-iteration $N \cdot T(n)$ times, and its error probability is bounded by e^{-N}.

Now we state the bound obtained by Schöning.

Theorem 1. For any formula F with n variables, if it is satisfiable, then the success probability of one repeat-iteration is at least $(3/4)^n$. Thus, the expected number of repeat-iterations executed until some satisfying assignment is found is at most $(4/3)^n$.

A key for proving this theorem is the following lemma, which will also play a key role in our discussion.

Lemma 2. For any satisfiable formula F, consider the execution of the above algorithm on F, and assume that a is selected as an initial assignment in some repeat-iteration. The probability that a satisfying assignment is found from this a is at least $(1/2)^d$, where d is the Hamming distance between a and some satisfying assignment a_* of F.

We show here how to prove Theorem 1 by using Lemma 2.

Proof of Theorem 1. Let F be any satisfiable formula with n variables, and let a_* be one of its satisfying assignments. (One can choose any satisfying assignment if F has more than one satisfying assignment.)

We estimate the probability Pr{sucess} that some satisfying assignment is found in one repeat-iteration. It follows from the lemma above that this probability can be calculated as follows. (We use $d(a, a_*)$ to denote the Hamming distance between a and a_*.)

$$\begin{aligned}
\Pr\{\text{sucess}\} &\geq \sum_{a\in\{0,1\}^n} \Pr\{\ a \text{ is selected as an initial assignment }\} \cdot \left(\frac{1}{2}\right)^{d(a,a_*)} \\
&= \sum_{d=0}^{n} \Pr\{\text{ the Hamming distance between } a \text{ and } a_* \text{ is } d\ \} \cdot \left(\frac{1}{2}\right)^{d}
\end{aligned}$$

On the other hand, the probability that an initial assignment with Hamming distance d from a_* is selected is $\binom{n}{d}(1/2)^d(1/2)^{n-d}$. Thus, we have by the binomial theorem

$$\begin{aligned}
\Pr\{\text{sucess}\} &\geq \sum_{d=0}^{n} \left(\binom{n}{d}\left(\frac{1}{2}\right)^{d}\left(\frac{1}{2}\right)^{n-d}\right) \cdot \left(\frac{1}{2}\right)^{d} \\
&= \sum_{d=0}^{n} \binom{n}{d}\left(\frac{1}{2}\cdot\frac{1}{2}\right)^{d}\left(\frac{1}{2}\right)^{n-d} = \left(\frac{1}{4}+\frac{1}{2}\right)^{n} = \left(\frac{3}{4}\right)^{n}.
\end{aligned}$$

Since the expected number of repeat-iterations is bounded by the inverse of the success probability Pr{sucess}, it is bounded by $(4/3)^n$. □

2 Independent Clauses and A Simple Deterministic Algorithm

Here we propose one simple deterministic algorithm for 3-SAT. We begin with introducing some notions.

For any clauses C and C', we say that C and C' are *independent* if no Boolean variable appears in both C and C'. For any formula F, a *maximal independent clause set* $\mathcal{C}$ is a set of clauses of F such that all clauses in $\mathcal{C}$ are (mutually) independent and every remaining clause of F shares some variable with some clause of $\mathcal{C}$.

Consider any formula F, and let $\mathcal{C}$ be any of its maximal independent clause set. Consider any partial assignment of F that assigns all and only variables in (clauses of) $\mathcal{C}$ so that all clauses in $\mathcal{C}$ are satisfied, and simplify F after fixing values of some variables by this partial assignment. (That is, we remove satisfied clauses from F and some false literals from some clauses of F.) Let $\widetilde{F}$ be the obtained formula. ($\widetilde{F}$ may become trivial, i.e., either constant 0 or 1, but in the following discussion, we assume that it is not trivial.) Then it is easy to see that each clause of $\widetilde{F}$ has at most two literals. This is because every remaining clause C' of $\widetilde{F}$ is obtained from some clause C not in $\mathcal{C}$, but C has some variable appearing in (some clause of) $\mathcal{C}$ that is assigned a value by the partial assignment. That is, $\widetilde{F}$ is a 2-CNF formula. Note, on the other hand, that 2-SAT is polynomial-time solvable [APT79]; that is, there is a polynomial-time algorithm for checking the satisfiability of a 2-CNF formula. This observation leads to a simple deterministic algorithm for 3-SAT stated as Figure 2. (For simplifying our later analysis, our algorithm is designed to choose clauses having exactly three variables for $\mathcal{C}$.)

Program Deterministic 3SAT Solver(**input** F);
  starting from the first clause of F,
    search for a clause C such that
    (1) C is independent with any clause in $\mathcal{C}$, and
    (2) C contains three variables,
    and add it to $\mathcal{C}$ until no such clause is found;
  for each satisfying assignment to clauses in $\mathcal{C}$ **do**
    $\widetilde{F} \leftarrow$ a 2-CNF formula obtained by simplifying F
        according to the assignment to variables in $\mathcal{C}$;
    if $\widetilde{F}$ is satisfiable **then accept** (and halt);
  end-for;
  reject;
end.

Figure 2: A simple deterministic algorithm

We can bound the computation time of this algorithm as follows.

Theorem 3. For any formula F with n variables and m clauses, consider the execution of the algorithm of Figure 2. Let $\widehat{m}$ be the number of independent clauses collected as $\mathcal{C}$. Then the algorithm runs in $\mathrm{poly}(n) \cdot 7^{\widehat{m}}$ steps.

Proof. For each clause C of $\mathcal{C}$, there are 7 ways to assign values to variables of C that satisfy C. Since all clauses in $\mathcal{C}$ are independent, we can assign each of ℓ clauses in $\mathcal{C}$ independently; hence, there are 7^{ℓ} satisfying assignments, and the algorithm stops after trying all of them. As explained above, after fixing the assignment to variables in $\mathcal{C}$, a formula F is simplified to a 2-CNF formula $\widetilde{F}$, for which we can decide satisfiability in $\mathrm{poly}(n)$ steps. Thus, the algorithm terminiates in $\mathrm{poly}(n) \cdot 7^{\widehat{m}}$ steps. $\square$

Clearly, the algorithm terminates faster if $\widehat{m}$ is smaller. But unfortunately, $\widehat{m}$ could be as large as $n/3$. In the next section, we propose an improvement of Schöning's algorithm that runs better for the case of $\widehat{m}$ being large. Then we will obtain our improved time bound by combining these two algorithms.

3 Our Improvement

In Schöning's algorithm, an initial assignment is computed by selecting each bit uniformly at random. Then roughly, the expected Hamming distance of the obtained initial assignment from a satisfying assignment (if it exists) becomes $n/2$. Intuitively, the success probability $\Pr\{\text{sucess}\}$ of one repeat-iteration gets better if we can improve the probability of starting from an assignment closer to a satisfying assignment. Here we propose a better way to compute an initial assignment, which assigns better values on variables in the maximal independent clause set.

Consider any satisfying formula F with n variables and m clauses, let it be fixed in the following discussion. Let $\widehat{m}$ be the number of independent clauses in $\mathcal{C}$ computed from F by our deterministic algorithm, and let $\widehat{n}$ be the number of variables (appearing) in (clauses of) $\mathcal{C}$. Since each clause in $\mathcal{C}$ has exactly three variables, we have $\widehat{n} = 3\widehat{m}$. By changing the sign of variables systematically, we can modify F to an equivalent one for which no negative literal appears in any clause of $\mathcal{C}$; that is, every clause C of $\mathcal{C}$ is of the form $C = (X_{i_1} \vee X_{i_2} \vee X_{i_3})$. In the following discussion, we assume the formula is modified in this way.

We explain our way to compute an initial assignment for F. For the analysis, it would be easier to determine the value of variables locally. The simplest way is to determine the value of each variable independently, which is actually approach taken in Schöning's algorithm. On the other hand, we can determine the value of variables in each clause in $\mathcal{C}$ independently, because clauses in $\mathcal{C}$ share no variables. Furthermore, by this way, we can exclude one assignment out of eight assignments. In fact, it is not necessary to select eight assignments with equal probability. Here we introduce three parameters x, y, and z to define the probability of selecting assignments for variables in each clause in $\mathcal{C}$. For variables not appearing in $\mathcal{C}$, we again assign 0 or 1 with probability 1/2.

More specifically, we propose the following selection procedure for determining an initial assignment of F.

- For each clause $C = (X_{i_1} \vee X_{i_2} \vee X_{i_3})$ in $\mathcal{C}$, select one of the eight assignments to $(X_{i_1}, X_{i_2}, X_{i_3})$ randomly according to the following probability:

assign	prob.	assign	prob.	assign	prob.
$(0,0,0)$	0	(1,1,0)	y	(0,0,1)	z
$(1,1,1)$	x	(1,0,1)	y	(0,1,0)	z
		(0,1,1)	y	(1,0,0)	z

- For each variable X_i not in in $\mathcal{C}$, select 0 or 1 randomly with probability 1/2.

Note that $(0,0,0)$ is excluded from our choice of an assignment for variables of any clause C in $\mathcal{C}$ because it does not satisfy C itself. Notice also that the parameters must satisfy $x+3y+3z = 1$.

Now we choose the values for the parameters x, y, and z so that a better initial assignment is selected with higher probability. More specifically, we would like to achieve the largest success probability. For this, we first examine how an initial assignment differs from a satisfying assignment, from which we estimate the success probablity Pr{sucess} of one repeat-iteration of Schöning's algorithm when an initial assignment is selected as above. Consider any satisfying assignment a_* of F, and let it be fixed in the following discussion.

Suppose, for example, that a_* assigns $(0,0,1)$ to variables of m' clauses in $\mathcal{C}$. Consider an initial assignment a that assigns $(0,0,1)$ to some of such clauses. Then a matches a_* on these clauses. On the other hand, if it assigns $(0,1,1)$ to some of m' clauses, then a differs from a_* by one bit on these clauses. The same situation also occurs if $(1,0,1)$ is selected for some of m' clauses when determining a. (Recall that $(0,0,0)$ is never selected.) In general, consider an initial assignment a in which s, t, u, and v clauses of those m' clauses are assigned values whose distances from the corresponding part of a_* are respectively, 0, 1, 2, and 3. (Let us call such an initial assignment as a (s,t,u,v)-*assignment*.) Note that there are $\binom{m'}{s,t,u,v} = \binom{m'}{s}\binom{m'-s}{t}\binom{m'-s-t}{u}$ ways of grouping m' clauses. On the other hand, we have the following probabilities:

$$
\begin{aligned}
&\Pr\{\ (0,0,1) \text{ is selected for } s \text{ clauses }\} \ = \ z^s,\\
&\Pr\{\ (0,1,1) \text{ or } (1,0,1) \text{ is selected for } t \text{ clauses }\} \ = \ (2y)^t,\\
&\Pr\{\ (1,1,1),\ (0,1,0), \text{ or } (1,0,0) \text{ is selected for } t \text{ clauses }\} \ = \ (x+2z)^u, \quad \text{and}\\
&\Pr\{\ (1,1,0) \text{ is selected for } t \text{ clauses }\} \ = \ y^v.
\end{aligned}
$$

Thus, the probability that some (s,t,u,v)-assignment is selected for a is

$$\binom{m'}{s,t,u,v} z^s (2y)^t (x+2z)^u y^v.$$

For any (s,t,u,v)-assignment a, the distance between a and a_* on m' clauses is $t+2u+3v$. As we will see later, the success probability can be calculated independently on independent clauses; that is, any (s,t,u,v)-assignment contributes $(1/2)^{t+2u+3v}$ to the success probability (on m' clauses). Hence, the total success probability (on m' clauses) of (s,t,u,v)-assignments become

$$\binom{m'}{s,t,u,v} z^s (2y)^t (x+2z)^u y^v \left(\frac{1}{2}\right)^{t+2u+3v}.$$

Therefore, by considering all possible grouping of m' clauses, we can estimate the total success probability (on m' clauses) as follows:

$$
\begin{aligned}
P'_1 &= \sum_{s+t+u+v=m'} \binom{m'}{s,t,u,v} z^s (2y)^t (x+2z)^u y^v \left(\frac{1}{2}\right)^{t+2u+3v}\\
&= \sum_{s+t+u+v=m'} \binom{m'}{s,t,u,v} z^s \left(\frac{2y}{2}\right)^t \left(\frac{x+2z}{4}\right)^u \left(\frac{y}{8}\right)^v\\
&= \left(z + \frac{2y}{2} + \frac{x+2z}{4} + \frac{y}{8}\right)^{m'} \ = \ \left(\frac{1}{4}x + \frac{9}{8}y + \frac{3}{2}z\right)^{m'}.
\end{aligned}
$$

A similar analysis yields the same result for the part that a_* assigns $(0,1,0)$ or $(1,0,0)$ to clauses in $\mathcal{C}$. Thus, letting m_1 be the number of clauses in $\mathcal{C}$ to which a_* assigns either $(0,0,1)$, $(0,1,0)$ or $(1,0,0)$, and P_1 be the success probability on m_1 clauses, we have

$$\begin{aligned}
P_1 &= \sum_{\substack{s_1+t_1+u_1\\+v_1=m_1}} \Pr\{\ a \text{ is a } (s_1,t_1,u_1,v_1)\text{-assignment }\} \cdot \left(\frac{1}{2}\right)^{t_1+2u_1+3v_1} \\
&= \left(\frac{1}{4}x+\frac{9}{8}y+\frac{3}{2}z\right)^{m_1}.
\end{aligned}$$

Let m_2 (resp., m_3) be the number of clauses in $\mathcal{C}$ to which a_* assigns either $(1,1,0)$, $(1,0,1)$ or $(0,1,1)$ (resp., $(1,1,1)$), and let P_2 and P_3 be the corresponding success probabilities. Then we similarly have

$$\begin{aligned}
P_2 &= \sum_{\substack{s_2+t_2+u_2\\+v_2=m_2}} \Pr\{\ a \text{ is a } (s_2,t_2,u_2,v_2)\text{-assignment }\} \cdot \left(\frac{1}{2}\right)^{t_2+2u_2+3v_2} \\
&= \sum_{\substack{s_2+t_2+u_2\\+v_2=m_2}} \binom{m_2}{s_2,t_2,u_2,v_2} y^{s_2}(x+2z)^{t_2}(2y)^{u_2}z^{v_2}\left(\frac{1}{2}\right)^{t_2+2u_2+3v_2} \\
&= \sum_{\substack{s_2+t_2+u_2\\+v_2=m_2}} \binom{m_2}{s_2,t_2,u_2,v_2} y^{s_2}\left(\frac{x+2z}{2}\right)^{t_2}\left(\frac{2y}{4}\right)^{u_2}\left(\frac{z}{8}\right)^{v_2} \\
&= \left(y+\frac{x+2z}{2}+\frac{2y}{4}+\frac{z}{8}\right)^{m_2} = \left(\frac{1}{2}x+\frac{3}{2}y+\frac{9}{8}z\right)^{m_2}. \\
P_3 &= \sum_{s_3+t_3+u_3=m_3} \Pr\{\ a \text{ is a } (s_3,t_3,u_3)\text{-assignment }\} \cdot \left(\frac{1}{2}\right)^{t_3+2u_3} \\
&= \binom{m_3}{s_3,t_3,u_3} x^{s_3}(3y)^{t_3}(3z)^{u_3}\left(\frac{1}{2}\right)^{t_3+2u_3} \\
&= \binom{m_3}{s_3,t_3,u_3} x^{s_3}\left(\frac{3y}{2}\right)^{t_3}\left(\frac{3z}{4}\right)^{u_3} \\
&= \left(x+\frac{3}{2}y+\frac{3}{4}z\right)^{m_3}.
\end{aligned}$$

Now we summarize our discussion and obtain the following bound for Pr{sucess}. (Recall that $\widehat{m}$ is the number of clauses in $\mathcal{C}$ and $\widehat{n}$ $(=3\widehat{m})$ is the number of variables in $\mathcal{C}$. Also note that $m_1+m_2+m_3=\widehat{m}$.)

Lemma 4. For any satisfiable formula F, consider the execution of Schöning's algorithm on F with our improved initial assignment selection procedure. Then we have

$$\Pr\{\text{sucess}\} \ \geq\ \left(\frac{3}{4}\right)^{n-\widehat{n}} \cdot \left(\frac{1}{4}x+\frac{9}{8}y+\frac{3}{2}z\right)^{m_1} \cdot \left(\frac{1}{2}x+\frac{3}{2}y+\frac{9}{8}z\right)^{m_2} \cdot \left(x+\frac{3}{2}y+\frac{3}{4}z\right)^{m_3}.$$

(The notations are the same in the above discussion. In particular, recall that $\widehat{m}$ is the number of clauses in $\mathcal{C}$ and $\widehat{n}$ $(=3\widehat{m})$ is the number of variables in $\mathcal{C}$. Also note that $m_1+m_2+m_3=\widehat{m}$.)

Proof. We start with the original bound explained in the proof of Theorem 1, which is

$$\Pr\{\text{sucess}\} \ \geq \sum_{a\in\{0,1\}^n} \Pr\{\ a \text{ is selected as an initial assignment }\} \cdot \left(\frac{1}{2}\right)^{d(a,a_*)}.$$

Consider any assignment a. It should be simultaneously a (s_1, t_1, u_1, v_1)-assignment, a (s_2, t_2, u_2, v_2)-assignment, and a (s_3, t_3, u_3)-assignment, for some $s_1, t_1, u_1, \ldots$. Then the distance $d(a, a_*)$ is calculated as $r + (t_1 + 2u_1 + 3v_1) + (t_2 + 2u_2 + 3v_2) + (t_3 + 2u_3)$, where r is the number of bits that a and a_* disagrees on variables not appearing in $\mathcal{C}$. (We use $V_{\overline{\mathcal{C}}}$ to denote the set of variables not appearing in $\mathcal{C}$.) This observation gives us the following restatement of the above bound.

$$\begin{aligned}
\Pr\{\text{sucess}\} \ \geq\ & \sum_{r=0}^{n-\widehat{n}} \sum_{\substack{s_1+t_1+u_1\\+v_1=m_1}} \sum_{\substack{s_2+t_2+u_2\\+v_2=m_2}} \sum_{s_3+t_3+u_3=m_3} \\
& \Pr\{\ a \text{ and } a_* \text{ disagree on } r \text{ bits of } V_{\overline{\mathcal{C}}}\ \} \times \Pr\{\ a \text{ is a } (s_1,t_1,u_1,v_1) \text{ assignment }\} \\
& \times \Pr\{\ a \text{ is a } (s_2,t_2,u_2,v_2)\text{-assignment }\} \times \Pr\{\ a \text{ is a } (s_3,t_3,u_3)\text{-assignment }\} \\
& \times \left(\frac{1}{2}\right)^{r+(t_1+2u_1+3v_1)+(t_2+2u_2+3v_2)+(t_3+2u_3)} \\
=\ & \left(\sum_{r=0}^{n-\widehat{n}} \Pr\{\ a \text{ and } a_* \text{ disagree on } r \text{ bits of } V_{\overline{\mathcal{C}}}\ \} \cdot \left(\frac{1}{2}\right)^{r}\right) \\
& \times \left(\sum_{\substack{s_1+t_1+u_1\\+v_1=m_1}} \Pr\{\ a \text{ is a } (s_1,t_1,u_1,v_1) \text{ assignment }\} \cdot \left(\frac{1}{2}\right)^{t_1+2u_1+3v_1}\right) \\
& \times \left(\sum_{\substack{s_2+t_2+u_2\\+v_2=m_2}} \Pr\{\ a \text{ is a } (s_2,t_2,u_2,v_2)\text{-assignment }\} \cdot \left(\frac{1}{2}\right)^{t_2+2u_2+3v_2}\right) \\
& \times \left(\sum_{s_3+t_3+u_3=m_3} \Pr\{\ a \text{ is a } (s_3,t_3,u_3)\text{-assignment }\} \cdot \left(\frac{1}{2}\right)^{t_3+2u_3}\right).
\end{aligned}$$

In the final expression, the second, the third, and the forth factors are respectively P_1, P_2, and P_3. On the other hand, the first factor is calculated in the same way as the proof of Theorem 1, and it is $(3/4)^{n-\widehat{n}}$. This gives us the desired bound. $\square$

Lemma 5. For our improved initial assignment selection procedure, to get the largest success probability (even in the worst-case) the parameters x, y, and z should be set as follows:

$$x \ = \ \frac{1}{7}, \ \ y \ = \ \frac{1}{7}\cdot\frac{2}{3} \ \ z \ = \ \frac{1}{7}\cdot\frac{4}{3}.$$

Then we have $\Pr\{\text{sucess}\} \geq (3/4)^{n-\widehat{n}}(3/7)^{\widehat{m}}$.

Proof. The lower bound given Lemma 4 depends on the choice of x, y, and z, and also $m_1 \sim m_3$; but we do not know $m_1 \sim m_3$ in advance. Thus, to get the best success probability even in the worst-case, we set the parameters x, y, and z to satisfy the following equations for some p:

$$\frac{1}{4}x + \frac{9}{8}y + \frac{3}{2}z \;=\; \frac{1}{2}x + \frac{3}{2}y + \frac{9}{8}z \;=\; x + \frac{3}{2}y + \frac{3}{4}z \;=\; p.$$

Solving these equation with the constraint $x+3y+3z = 1$, we obtain the values of the parameters given in the lemma and $p = 3/7$. $\square$

We summarize our discussion.

Theorem 6. For any formula F with n variables and m clauses, let $\widehat{m}$ be the number of independent clauses collected as $\mathcal{C}$ in our deterministic algorithm. If F is satisfiable, then the expected running time of our improved randomized algorithm on F is at most $\mathrm{poly}(n)((4/3)^{n-3\widehat{m}}(7/3)^{\widehat{m}})$.

Finally, we estimate a time bound when executing our two algorithms together. It follows from Theorem 3 and Theorem 6, the expected running time of the combined algorithm is bounded by the minimum of

$$\mathrm{poly}(n) \cdot 7^{\widehat{m}} \quad \text{and} \quad \mathrm{poly}(n) \cdot (4/3)^{n-3\widehat{m}}(7/3)^{\widehat{m}}.$$

Hence, the asymptotically worst-case is the case where

$$7^{\widehat{m}} \;=\; (4/3)^{n-3\widehat{m}}(7/3)^{\widehat{m}},$$

which is the case where $\widehat{m} \approx 0.146652n$. Even in this case, the running time is bounded by $7^{0.146652n} \le 1.3303^n$.

Therefore, we have the following theorem.

Theorem 7. For any formula F with n variables and m clauses, if F is satisfiable, then the expected running time of the combined algorithm on F is at most $\mathrm{poly}(n) \cdot (1.3303)^n$.

References

[APT79] B. Aspvall, M.F. Plass, R.E. Tarjan, A linear-time algorithm for testing the truth of certain quantified Boolean formulas, *Information Processing Letters*, 8(3), 121–123, 1979.

[DGHS00] E. Dantsin, A. Goerdt, E.A. Hirsch, and U. Schöning, Deterministic algorithms for k-SAT based on covering codes and local search, in *Proc. 27th International Colloquium on Automata, Languages and Programming*, ICALP'00, Lecture Notes in Comp. Sci. 1853, 236–247, 2000.

[Hir00a] E.A. Hirsch, New worst-case upper bounds for SAT, *Journal of Automated Reasoning*, 24(4), 397–420, 2000.

[Hir00b] E.A. Hirsch, SAT local search algorithms: worst-case study, *Journal of Automated Reasoning*, 24(1/2), 127–143, 2000.

[KP92] E. Koutsoupias and C.H. Papadimitriou, On the greedy algorithm for satisfiability, *Information Processing Letters*, 43(1), 53–55, 1992.

[Sch99] U. Schöning, A probabilistic algorithm for k-SAT and constraint satisfaction problems, in *Proc. of the 40th Ann. IEEE Sympos. on Foundations of Comp. Sci.* (FOCS'99), IEEE, 410–414, 1999.

[PPSZ98] R. Paturi, P. Pudlák, M.E. Saks, and F. Zane, An improved exponential-time algorithm for k-SAT, in *Proc. of the 39th Ann. IEEE Sympos. on Foundations of Comp. Sci.* (FOCS'98), IEEE, 628–637, 1998.

[MS85] B. Monien and E. Speckenmeyer, Solving satisfiability in less than 2^n steps, *Discrete Applied Mathematics*, 10, 287–295, 1985.

[Kul99] O. Kullmann, New methods for 3-SAT decision and worst-case analysis, *Theoretical Computer Science*, 223(1/2), 1–72, 1999.