Overview

Local search paradigm

pick a total assignment α
repeat
    if $\alpha \models F$ then return α
    pick a variable x
    $\alpha := \alpha$ with $[x := 1 - \alpha(x)]$

Both choices can be either random or deterministic:

- α random, x deterministic
  $\rightsquigarrow$ Hamming ball algorithm
- both deterministic
  $\rightsquigarrow$ derandomized Hamming ball algorithm
- both random
  $\rightsquigarrow$ Schöning's random walk

Hamming balls

For total assignments α, β, the Hamming distance is

$$d(\alpha, \beta) = |\{x; \alpha(x) \neq \beta(x)\}|$$

The Hamming ball of radius r around α is

$$H(\alpha, r) := \{\beta; d(\alpha, \beta) \leq r\}$$

The size of a Hamming ball is

$$V(r) := |H(\alpha, r)| = \sum_{i=0}^{r} \binom{n}{i}$$

Theorem

Let $r := \rho n$, and let $h(\rho) = -\rho \log \rho - (1-\rho)\log(1-\rho)$ be the entropy function.

$$\frac{1}{\sqrt{8n\rho(1-\rho)}} 2^{h(\rho)n} \leq V(r) \leq 2^{h(\rho)n}$$

Searching a Hamming ball

This procedure searches a Hamming ball around α for an assignment satisfying F:

search(α, r)
  if $\alpha \models F$
    then return α
  if $r = 0$
    then return NULL
  pick $C = a_1 \vee \ldots \vee a_k$ s.t. $C\alpha = 0$
  for $i := 1$ to k do
    $\alpha_i := \alpha$ with $[a_i \leftarrow 1]$
    $\beta :=$ search$(\alpha_i, r-1)$
    if $\beta \neq$ NULL
      then return β
  return NULL

Analysis of `search`

Run-time of $\texttt{search}(\alpha, r)$ is k^r.

Possibly $k^r < V(r)$, e.g. for $k = 3, r = n/2$:

$$V(r) \geq 2^{n-1}, \quad \text{but } k^r = 3^{n/2} = \sqrt{3}^n \leq 1.7321^n$$

Simple algorithm for 3-SAT:

$\texttt{search}(\vec{0}, n/2)$
$\texttt{search}(\vec{1}, n/2)$

This algorithm solves 3-SAT in time $O(\sqrt{3}^n) = O(1.7321^n)$.

Hamming ball algorithm

Smaller radius $r = \rho n$ for some $\rho < 1/2$,
more randomly selected starting points:

```
for i := 1 to w do
      pick α_i randomly
      β := search(α_i, ρn)
      if β ≠ NULL
         then return β
  return NULL
```

Run-time is $w \cdot k^{\rho n}$

$\rightsquigarrow$ choose parameters w and ρ to minimize error probability.

Analysis of the Hamming ball algorithm

Let $\alpha \models F$. $\quad$ `search`$(\alpha_i, \rho n)$ finds α, if $\alpha_i \in H(\alpha, \rho n)$.

This happens with probability $V(\rho n)2^{-n} \geq 2^{-(1-h(\rho))n}$.

Probability that algorithm does not find α is bounded by

$$(1 - 2^{-(1-h(\rho))n})^w \leq e^{-w2^{-(1-h(\rho))n}}.$$

Need $w = c \cdot 2^{(1-h(\rho))n}$ repetitions to make error $< e^{-c}$.

Thus: run-time is $2^{(1-h(\rho))n} k^{\rho n} = (2\rho^\rho (1-\rho)^{(1-\rho)} k^\rho)^n$

The basis is minimized for $\rho = 1/(k+1)$,
giving run-time $(\frac{2k}{k+1})^n$.

k	3	4	5	6	7	8
$\frac{2k}{k+1}$	1.5	1.6	1.6667	1.7143	1.75	1.7778

Covering codes

A set C of assignments is a covering code of radius r,
if every assignment $\beta \in H(\alpha, r)$ for some $\alpha \in C$.

Covering code yields deterministic algorithm of run-time $|C| \cdot k^r$:

```
for each γ ∈ C do
    β := search(γ, r)
    if β ≠ NULL
        then return β
return NULL
```

Thus: need deterministic construction of covering codes.

The Set Cover problem

Covering code: special case of Set Cover problem:

Instance:	A set S, and a family $F \subseteq 2^S$ of subsets of S
Solution:	A subfamily $C \subseteq F$ covering S, i.e. $S = \bigcup C$.
Objective:	Minimize $\lvert C \rvert$.

Here S is the set of assignments,

and F the set of Hamming balls of radius r.

Greedy algorithm for Set Cover

Greedy set cover:

$C := \emptyset$
`while` $S \not\subseteq \bigcup C$
  `pick` $c \in F$ `s.t.` $c \setminus \bigcup C$ `maximal`
  $C := C \cup \{c\}$

Lemma

The greedy algorithm approximates Set Cover *to a factor of* $\log |S|$.

I.e., if C^* is a minimal set cover, then the algorithm computes a cover C with $|C| \leq |C^*| \cdot \log |S|$.

Here $|S| = 2^n$, so the factor $\log |S| = n$.

Existence of covering codes

Lemma

For $r = \rho n$ with $0 < \rho \leq 1/2$,
there is a covering code of radius r of size

$$s \leq n \cdot \sqrt{8n\rho(1-\rho)} \cdot 2^{(1-h(\rho))n}$$

Thus: greedy algorithm finds a covering code of size

$$n \cdot s = O(n^{5/2}) \cdot 2^{(1-h(\rho))n}.$$

But: run-time is about 2^{3n}.

Boosting a covering code

Lemma

C a covering code for n variables of radius r

$\rightsquigarrow$ *covering code C^d for dn variables of radius dr, of size $|C|^d$.*

Easy application:

- Use greedy to compute code C for $n/6$ variables of radius $\rho n/6$. This takes time $2^{3n/6} = \sqrt{2}^n$.
- Use the code C^6 of size $|C^6| = O(n^{15}) \cdot 2^{(1-h(\rho))n}$.
- run-time is essentially $(2^{(1-h(\rho))} k^\rho)^n$
  $\rightsquigarrow$ same as probabilistic Hamming ball algorithm.
- But: Greedy algorithm requires exponential space.

Improved derandomized algorithm

Slightly slower algorithm with low space usage:

- For a $\delta > 0$, find the least b with
$$b \cdot \sqrt{8b\rho(1-\rho)} \cdot 2^{(1-h(\rho))b} \leq 2^{(1-h(\rho)+\delta)b}$$
- By the lemma there is a code C_b for b variables of radius ρb, of size $2^{(1-h(\rho)+\delta)b}$.
- Such a code C_b - found by exhaustive search - is hard-wired.
- Algorithm uses the code $(C_b)^{n/b}$ of size $2^{(1-h(\rho)+\delta)n}$.
- For $\epsilon > 0$, let $\delta := \log(\frac{k+1}{2k}\epsilon + 1)$.
- With this choice of δ, the run-time is $(\frac{2k}{k+1} + \epsilon)^n$.

Local search paradigm

pick a total assignment α
repeat
    if $\alpha \models F$ then return α
    pick a variable x
    $\alpha := \alpha$ with $[x := 1 - \alpha(x)]$

Both choices can be either random or deterministic:

- α random, x deterministic
  $\rightsquigarrow$ Hamming ball algorithm
- both deterministic
  $\rightsquigarrow$ derandomized Hamming ball algorithm
- both random
  $\rightsquigarrow$ Schöning's random walk

Schöning's random walk algorithm

repeat w times

  pick α randomly

  repeat $3n$ times

    if $\alpha \models F$

      then return α

    pick $C = a_1 \vee \ldots \vee a_k$ with $C\alpha = 0$

    pick i randomly from $\{1, \ldots, k\}$

    $\alpha := \alpha$ with $[a_i \leftarrow 1]$

return NULL

Success probability and run-time

Let $\alpha^* \models F$ be fixed.

We estimate the success probability $p(n)$
i.e., the probability that the inner loop finds α^*.

Then the error probability is $(1 - p(n))^w \leq e^{-wp(n)}$

$\rightsquigarrow$ $w = c\frac{1}{p(n)}$ repetitions to make error $< e^{-c}$.

Estimating the Hamming distance

For every clause C, fix $a_C \in C$ with $\alpha^*(a_C) = 1$.

Define a random variable Z:

- In the beginning set $Z := d(\alpha, \alpha^*)$.
- After iteration when C and $a_i \in C$ were picked, update Z:

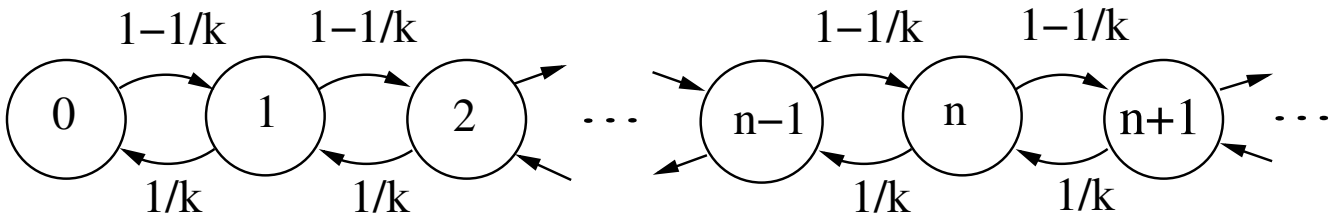
$$Z := \begin{cases} Z - 1 & \text{if } a_i = a_C \\ Z + 1 & \text{otherwise.} \end{cases}$$

After every iteration, we have $Z \geq d(\alpha, \alpha^*)$.

Thus, probability that α^* is found $\geq$ probability that $Z = 0$ reached.

Markov chain

The random variable Z is a Markov chain:



Z is initially in a random state i with $0 \leq i \leq n$, and in each step

- decremented with probability $1/k$ and
- incremented with probability $1 - 1/k$.

Initial state is i with probability $\binom{n}{i} 2^{-n}$.

Probability that state 0 is reached from state i:

$p_i := (\frac{1}{k-1})^i$

Analysis of the random walk algorithm

The probablity of reaching $Z = 0$ is thus

$$\begin{aligned} & \sum_{i=0}^{n} 2^{-n} \binom{n}{i} p_i \\ = \quad & 2^{-n} \sum_{i=0}^{n} \binom{n}{i} (\frac{1}{k-1})^i \\ = \quad & 2^{-n} (1 + \frac{1}{k-1})^n \\ = \quad & (\frac{k}{2(k-1)})^n \end{aligned}$$

Thus: run-time of Schöning's random walk for k-SAT is $\left(\frac{2(k-1)}{k}\right)^n$.

For small k, the basis is:

k	3	4	5	6	7	8
$\frac{2(k-1)}{k}$	1.33334	1.5	1.6	1.66667	1.71429	1.75

Improvement by independent clauses

Hofmeister, Schöning, Schuler and Watanabe improve
the random walk algorithm for 3-SAT by using independent clauses.

Definition

A set $G \subseteq F$ of clauses is independent,
if $V(C) \cap V(D) = \emptyset$ for any $C \neq D \in G$.

If G is independent, then $|V(G)| = 3|G|$.

First, use a greedy algorithm to compute a maximal independent set:

$G := \emptyset$
`while there is` $C \in F$ `with` $V(C) \cap V(G) = \emptyset$
    $G := G \cup \{C\}$

First case: $|G|$ is small

G maximally independent:

$\rightsquigarrow$ every clause in $F \setminus G$ contains a variable in $V(G)$.

Let $0 < \delta \leq 1$ be a parameter (to be determined).

If $|G| \leq \delta n$:

- There are $7^{|G|} \leq 7^{\delta n}$ assignments $\alpha : V(G) \to \{0, 1\}$ with $\alpha \models G$.
- For each such α, solve the 2-SAT formula $F\alpha$ in polynomial time.
- Run-time is $7^{\delta n}$.

Second case: $|G|$ is large

If $|G| > \delta n$:

Use the random walk algorithm, with starting points picked according to a distribution $V_G(p_1, p_2, p_3)$:

First, for each $C = a_1 \vee a_2 \vee a_3$ in G, pick $\epsilon_1, \epsilon_2, \epsilon_3$
with probabilities according to:

ϵ_1	ϵ_2	ϵ_3		ϵ_1	ϵ_2	ϵ_3	
0	0	0	0	0	1	1	p_2
0	0	1	p_1	1	0	1	p_2
0	1	0	p_1	1	1	0	p_2
1	0	0	p_1	1	1	1	p_3

and set $[a_1 := \epsilon_1, a_2 := \epsilon_2, a_3 := \epsilon_3]$.

Then for each $x \in V(F) \setminus V(G)$ pick uniformly a value $\epsilon \in \{0, 1\}$
and set $[x := \epsilon]$.

Alternative analysis of the random walk

Recall: Probabilty that $\alpha^* \models F$ is reached from α is at least $(1/2)^{d(\alpha,\alpha^*)}$.

Let $p(\alpha)$ be the probabilty that α is picked as starting point.

$$p(n) \;\geq\; \sum_\alpha p(\alpha)\left(\tfrac{1}{2}\right)^{d(\alpha,\alpha^*)} \;=\; E[\left(\tfrac{1}{2}\right)^{d(\alpha,\alpha^*)}]$$

Let random variable X_i be the indicator of $\alpha(x_i) \neq \alpha^*(x_i)$.

Thus $d(\alpha, \alpha^*) = X_1 + \ldots + X_n$.

When α picked uniformly $\rightsquigarrow$ X_i independent:

$$E\left[\left(\tfrac{1}{2}\right)^{d(\alpha,\alpha^*)}\right] \;=\; E\left[\left(\tfrac{1}{2}\right)^{X_1+\ldots+X_n}\right] \;=\; \prod_{i=1}^{n} E\left[\left(\tfrac{1}{2}\right)^{X_i}\right]$$

$E\left[\left(\tfrac{1}{2}\right)^{X_i}\right] = 3/4$, thus $p(n) \geq (3/4)^n$.

Estimating the success probability

α picked according to $V_G(p_1, p_2, p_3) \rightsquigarrow X_i$ not independent.

Let $G = \{C_1, \ldots, C_{\delta n}\}$ with $V(C_{i+1}) = \{x_{3i+1}, x_{3i+2}, x_{3i+3}\}$,
and let random variable $S_i = X_{3i+1} + X_{3i+2} + X_{3i+3}$.

S_i are independent:
$E\left[\left(\frac{1}{2}\right)^{X_1+\ldots+X_n}\right] = \prod_{i=1}^{\delta n} E\left[\left(\frac{1}{2}\right)^{S_i}\right] \cdot \prod_{j=3\delta n+1}^{n} E\left[\left(\frac{1}{2}\right)^{X_j}\right]$

The right factor $\prod_{j=3\delta n+1}^{n} E\left[\left(\frac{1}{2}\right)^{X_j}\right] = \left(\frac{3}{4}\right)^{(1-3\delta)n}$

Estimating the success probability

Expectation of $\left(\frac{1}{2}\right)^{S_i}$ depends on the number of literals in C_i satisfied by α^*.

- α^* satisfies 1 literal in C_i: $E\left[\left(\frac{1}{2}\right)^{S_i}\right] = \frac{3}{2}p_1 + \frac{9}{8}p_2 + \frac{1}{4}p_3$.
- α^* satisfies 2 literals in C_i: $E\left[\left(\frac{1}{2}\right)^{S_i}\right] = \frac{9}{8}p_1 + \frac{3}{2}p_2 + \frac{1}{2}p_3$.
- α^* satisfies 3 literals in C_i: $E\left[\left(\frac{1}{2}\right)^{S_i}\right] = \frac{3}{4}p_1 + \frac{3}{2}p_2 + p_3$.

Let m_d be the number of clauses in G where α^* satifies d literals.

The term $\prod_{i=1}^{\delta n} E\left[\left(\frac{1}{2}\right)^{S_i}\right]$ is

$$\left(\frac{3}{2}p_1 + \frac{9}{8}p_2 + \frac{1}{4}p_3\right)^{m_1} \cdot \left(\frac{9}{8}p_1 + \frac{3}{2}p_2 + \frac{1}{2}p_3\right)^{m_2} \cdot \left(\frac{3}{4}p_1 + \frac{3}{2}p_2 + p_3\right)^{m_3}$$

Product gets largest when the three bases are equal.

Calculating the run-time

Equality of the bases, together with $3p_1 + 3p_2 + p_3 = 1$, yields:
$p_1 = \frac{4}{21}$, $p_2 = \frac{2}{21}$, $p_3 = \frac{1}{7}$

These give the bases as 3/7, thus $\prod_{i=1}^{\delta n} E\left[\left(\frac{1}{2}\right)^{S_i}\right] = \left(\frac{3}{7}\right)^{\delta n}$.

Success probability in this case is $p(n) \geq \left(\frac{3}{7}\right)^{\delta n} \cdot \left(\frac{3}{4}\right)^{(1-3\delta)n}$

Run-time in this case is $\left(\left(\frac{7}{3}\right)^{\delta} \cdot \left(\frac{4}{3}\right)^{(1-3\delta)}\right)^{n}$

Monotone decreasing in δ, run-time $7^{\delta n}$ in other case increasing.

Minimal at intersection where $\left(\frac{7}{3}\right)^{\delta} \cdot \left(\frac{4}{3}\right)^{(1-3\delta)} = 7^{\delta}$.

This yields $\delta = 01.4666$, where $7^{\delta} \leq 1.33026$.

Thus the run-time is 1.33026^{n}.

Searching Hamming balls and run-time

Analysis of Hamming ball algorithm shows:

Hamming ball of radius r searched in time t^r

$\rightsquigarrow$ (deterministic) algorithm for k-SAT in time $(\frac{2t}{t+1})^n$

Now: algorithm `MS-search`(α, r) to deterministically search Hamming ball of radius r in time $O((k-1+\delta)^r)$

$\rightsquigarrow$ deterministic algorithm for k-SAT in time $(\frac{2(k-1)}{k} + \epsilon)^n$

Property: If there is $\alpha^* \models F$ with $d(\alpha, \alpha^*) \leq r$,
then `MS-search`(α, r) returns a satisfying assignment $\alpha \models F$.

k-ary covering codes

For $w = (w_1, \ldots, w_t), w' \in \{1, \ldots, k\}^t$ define

the Hamming distance $\quad d(w, w') := |\{\, i \,;\, w_i \neq w'_i \,\}|$

Hamming ball of radius r: $\quad H_k(w, r) := \{w' \,;\, d(w', w) \leq r\}$

$V_k(t, r) := |H_k(w, r)| \geq \binom{t}{r}(k-1)^r$

A set $C \subseteq \{1, \ldots k\}^r$ is a covering code of radius r,

if every $w' \in H_k(w, r)$ for some $w \in C$.

Existence k-ary covering codes

Lemma

For $0 \leq r \leq t$, there is a covering code $C \subseteq \{1, \ldots, k\}^t$ of radius r

of size $$|C| \leq \frac{t \ln(k) k^t}{\binom{t}{r}(k-1)^r} \quad \leq \quad t^2 (k-1)^{t-2t/k}$$

The algorithm uses a covering code $C \leq \{1, \ldots, k\}^t$ of radius t/k for a sufficiently large constant t.

This code C of size $|C| \leq t^2(k-1)^{t-2t/k}$ is pre-computed and hard-wired.

Algorithm of Moser and Scheder: easy case

Pick a maximal independent set G of independent clauses

$$G = \{C_1, \ldots, C_m\} \text{ with } \alpha(C_i) = 0.$$

Case 1: $m < t$:

- For each $\beta : V(G) \to \{0,1\}$ with $\beta \models G$, run $\texttt{search}(\alpha, r)$ on $F\beta$.
- Since G is maximally independent, $F\beta$ is a $(k-1)$-CNF for every β.
- Therefore the run-time is $2^{km} \cdot (k-1)^r = O((k-1)^r)$.

Algorithm of Moser and Scheder: hard case

Case 2: $m \geq t$

Pick a subset $G' = \{C_1, \ldots, C_t\} \subseteq G$ of size $|G'| = t$,
where $C_i = a_{i,1} \vee \ldots \vee a_{i,k}$.

For $w \in \{1, \ldots, k\}^t$, let $\alpha[w]$ be the assignment
α `with` $[\, a_{1,w_1} := 1 - \alpha(a_{1,w_1}) \,, \ldots \,, a_{k,w_k} := 1 - \alpha(a_{k,w_k}) \,]$

Pick w^* such that $\alpha^*(a_{i,w_i^*}) = 1$ for $i = 1, \ldots, t$.
Thus $d(\alpha[w^*], \alpha^*) = d(\alpha, \alpha^*) - t \leq r - t$.

Algorithm of Moser and Scheder: run-time

For some $w \in C$: $\quad d(w, w^*) \leq t/k$

$\rightsquigarrow \quad d(\alpha[w], \alpha^*) \;\leq\; d(\alpha, \alpha^*) + t/k - (t - t/k) \;\leq\; r - (t - 2t/k)$

Let $\Delta := (t - 2t/k)$. For every $w \in C$, recursively call

$\texttt{MS-search}(\alpha[w], r - \Delta)$

$|C|$-branching search tree of depth r/Δ, of size

$$|C|^{r/\Delta} \leq \left(t^2(k-1)^{\Delta}\right)^{r/\Delta} = \left((k-1)t^{2/\Delta}\right)^r$$

Pick t sufficiently large such that $(k-1)t^{2/\Delta} < (k-1) + \delta$.

$\rightsquigarrow \quad$ run-time is $O\left((k - 1 + \delta)^r\right)$

Algorithm of Paturi, Pudlák and Zane

Let $\pi \in S_n$ be a permutation of the variables.

search(F, π)

  $\alpha := []$

  for $i := 1$ to n do

    if unit clause $x^{\epsilon}_{\pi(i)}$ in $F\alpha$

      then $\epsilon_i := \epsilon$

      else pick ϵ_i randomly from $\{0, 1\}$

    $\alpha = \alpha \cup [x_{\pi(i)} \leftarrow \epsilon_i]$

  if $\alpha \models F$

    then return α

    else return NULL

Algorithm of Paturi, Pudlák and Zane

```
PPZ(F)
  repeat w times
     pick π ∈ S_n uniformly random
     β := search(F, π)
     if β ≠ NULL
        then return β
  return NULL
```

Lower bound $p(n)$ on the success probability

$\rightsquigarrow$ upper bound $O(1/p(n))$ on the number of repetitions, and thus the run-time.

Critical variables

For a set A of assignments, let $N_A(\alpha) := |H(\alpha, 1) \cap A| - 1$.

Lemma

If $A \neq \emptyset$, then $\sum_{\alpha \in A} 2^{-N_A(\alpha)} \geq 1$.

Let $A := \{\alpha\,;\ \alpha \models F\}$, and fix $\alpha \in A$.

x_i is critical if for $\alpha_i := \alpha$ `with` $[x_i := 1 - \alpha(x_i)]$, $\quad \alpha_i \not\models F$.

There are $j(\alpha) := n - N_A(\alpha)$ critical variables.

Critical clauses

For every critical x_i there is a critical clause C_i,

s.t. x_i (or $\bar{x}_i$) is the only satisfied literal in C_i.

$R(\pi, \alpha) := \left\{ x_i \text{ critical} ; \pi^{-1}(i) \geq \pi^{-1}(j) \text{ for all } x_j \in V(C_i) \right\}$

$r(\pi, \alpha) := |R(\pi, \alpha)|$.

π picked uniformly, thus $E[r(\pi, \alpha)] = j(\alpha)/k$

Probability $p_{\alpha,\pi}$ that α is found, given that π was picked:

$p_{\alpha,\pi} \geq 2^{-n+r(\pi,\alpha)}$

The probability to find an assignment

Probability p_α that α is found:

$$\begin{aligned}
p_\alpha &= \sum_{\pi \in S_n} \frac{1}{n!} p_{\alpha,\pi} \\
&\geq \sum_{\pi \in S_n} \frac{1}{n!} 2^{-n+r(\pi,\alpha)} \\
&= 2^{-n} \sum_{\pi \in S_n} \frac{1}{n!} 2^{r(\pi,\alpha)} \\
&= 2^{-n} E[2^{r(\pi,\alpha)}] \\
&\geq 2^{-n+E[r(\pi,\alpha)]} \qquad (*) \\
&\geq 2^{-n+j(\alpha)/k}
\end{aligned}$$

$(*)$ holds due to Jensen's inequality:

If X takes n values each with prob. $1/n$, then $E[2^X] \geq 2^{E[X]}$

Calculating the run-time

Thus the success probability is:

$$\begin{aligned} p(n) &= \sum_{\alpha\in A} p_\alpha \geq \sum_{\alpha\in A} 2^{-n+j(\alpha)/k} \\ &= \sum_{\alpha\in A} 2^{-n+n/k+(j(\alpha)-n)/k} \\ &= 2^{-n+n/k} \sum_{\alpha\in A} 2^{-N_A(\alpha)/k} \\ &\geq 2^{-n+n/k} \sum_{\alpha\in A} 2^{-N_A(\alpha)} \\ &\geq 2^{-n+n/k} \end{aligned}$$

Therefore, the run-time is $2^{n-n/k} = \left(2^{1-1/k}\right)^n$. The basis for small k is:

k	3	4	5	6	7	8
$2^{1-1/k}$	1.58741	1.68180	1.74111	1.78180	1.81145	1.83401

Algorithm of Paturi, Pudlák, Saks and Zane

Paturi, Pudlák, Saks and Zane improve the PPZ algorithm with a preprocessing step.

PPSZ(F, s)
  $F_s := F$
  while there are $C, C' \in F_s$ such that
      C, C' clash
      and $D := Res(C, C') \notin F_s$
      and $w(D) \leq s$
    $F_s := F_s \wedge D$
  return PPZ(F_s)

The run-time of PPSZ for k-SAT is b_k^n, where b_k for small k is

k	3	4	5	6	7	8
b_k	1.36406	1.49579	1.56943	1.63788	1.68753	1.72521