A Full Derandomization of Schöning's k-SAT Algoritmh

by *Robin A. Moser and Dominik Scheder*
presented by Dušan Knop

Promise Ball-k-SAT Given a k-CNF formula φ over n variables, an assignment a, a natural number r, and the promise that $B_r(a)$ contains a satisfying assignment. Find any satisfying assignment to φ.

Schöning's Algorithm The algorithm is very simple – consider a probabilistic procedure with k-CNF formula on input that guesses an initial assignment $a \in \{0,1\}^n$, uniformly at random. Then it repeats $3n$ times let C be a clause unsatisfied by actual assignment and pick one of its literals in the clause at random and flip its value in the current assignment.

Suppose we have a satisfiable formula and fix some satisfying assignment a^*. We want to estimate the probability p that the algorithm finds a^* (or other satisfying assignment). Note that Hamming distance to a^* is important for analysis of the procedure. If C is an unsatisfied clause then there is at least one literal (out of at most k) that decreases Hamming distance to a^* – so from state with distance j transfers to state $j-1$ with probability at least $1/k$ (and to $j+1$ with probability at most $(k-1)/k$). Procedure starts Markov chain and terminates after at most $3n$ steps.

Given that the process has initially transfered into state j we calculate the probability g_j that the process reaches the absorbing state 0. We consider the case that the process takes $i \leq j$ steps in the "wrong" direction (then $i+j$ steps must be done in the "right" direction). Please observe the similarity with counting number of paths in rectangular grid – using ballot theorem it is $\binom{j+2i}{i} \cdot \frac{j}{j+2i}$.

$$\begin{aligned} g_j &\geq \frac{1}{3}\sum_{i=0}^{j} \binom{j+2i}{i} \left(\frac{k-1}{k}\right)^i \left(\frac{1}{k}\right)^{i+j} \geq \\ &\geq \left[\left(\frac{1+2\alpha}{\alpha}\right)^\alpha \left(\frac{1+2\alpha}{1+\alpha}\right)^{1+\alpha} \left(\frac{k-1}{k}\right)^\alpha \left(\frac{1}{k}\right)^{1+\alpha}\right]^j \geq \left(\frac{1}{k-1}\right)^j \end{aligned}$$

Using this result we can calculate the probability of success of the procedure p:

$$p \geq \left(\frac{1}{2}\right)^n \sum_{j=0}^{n} \binom{n}{j} \left(\frac{1}{k-1}\right)^j = \left(\frac{1}{2}\left(1 + \frac{1}{k-1}\right)\right)^n.$$

Notice that we needed to consider random walks up to length $j + 2i \leq n + 2n = 3n$ only.

Lemma 1 (Dantsin et al.)**.** *If algorithm A solves Promise Ball-k-SAT in time $O^*(\alpha^r)$, then there is algorithm solving k-SAT in time $O^*((\frac{2\alpha}{\alpha+1})^n)$. Furthermore this algorithm is deterministic if A is.*

Ingredient: k-ary Covering Codes Let $t \in \mathbb{N}$. A set $\mathcal{C} \subseteq \{1,\ldots,k\}^t$ is called a code of covering radius r if $\cup_{w\in\mathcal{C}} B_r^{(k)} = \{1,\ldots,k\}$.

Lemma 2. *For any $t,k \in \mathbb{N}$ and $0 \leq r \leq t$, there is a code $\mathcal{C} \subseteq \{1,\ldots,k\}^t$ of covering radius r such that $|\mathcal{C}| \leq \lceil \frac{t\ln(k)k^t}{\binom{t}{r}(k-1)^r} \rceil$.*

We will now describe the deterministic algorithm. First it chooses a sufficiently large constant t, depending on the ε, and computes a code $\mathcal{C} \subseteq \{1, \ldots, k\}^t$ of covering radius t/k. Since k and t are constants, it can afford to compute an optimal such code. We estimate its size: $\mathcal{C} \leq t^2(k-1)^{t-2t/k}$. So the code $\mathcal{C}$ is constant sized, can be computed and stored for further use.

First of all: Construct greedily a maximal set G of pairwise disjoint unsatisfied k-clauses of φ. That is $G = \{C_1, \ldots, C_m\}$, the C_i are pairwise disjoint and unsatisfied by assignment a and each unsatisfied k-clause D in φ shares at least one literal with some C_i.

First Case ($m < t$): enumerate all 2^{km} truth assignments to the variables of G and fix this values–note that this reduces the size of all k-clauses by 1, and so the exhaustive search through the ball $B_r(a)$ take running time $O^*((k-1)^r)$. Since t is constant $2^{km}O^*((k-1)^r) = O^*((k-1)^r)$.

Second Case ($m \geq t$): Choose t clauses from G to form $H = \{C_1, \ldots, C_t\}$. For $w \in \{1, \ldots, k\}$ let $a[w]$ be the assignment obtained from a by flipping w_i-th literal in clause C_i. Consider now promised satisfying assignment a^* with $d(a, a^*) \leq r$ and define w^* as follows: for each $1 \leq i \leq t$, we set w_i^* to j such that a^* satisfies j-th literal in C_i–note that $d(a[w^*], a^*) \leq r - t$.

We could iterate over all $w \in \{1, \ldots, k\}$ without using the flavor of Covering Codes–but this would yield a running time of $O^*(k^r)$. Rather we add the flavor and iterate only over $w \in \mathcal{C}$–by properties of $\mathcal{C}$ there is $w' \in \mathcal{C}$ with $d(w', w^*) \leq t/k$ (steps in bad direction). Therefore $d(a[w'], a^*) \leq d(a, a^*) + t/k - (t - t/k) \leq r - (t - 2t/k)$.

Set $\Delta := (t - 2k/t)$ and use recursion with $a[w]$ and $r - \Delta$ for each $e \in \mathcal{C}$–number of leaves in recursion is at most $|\mathcal{C}|^{r/\Delta} \leq (t^2(k-1^{\Delta}))^{r/\Delta} = ((k-1)^{t^2/\Delta})^r$. Since t^2/Δ goes to 1 as t grows, the above term is bounded by $(k - 1 + \varepsilon)$ (for sufficiently large t).

Theorem 1. *For every $\varepsilon > 0$, there exists a deterministic algorithm which solves the Promise Ball-k-SAT problem in time $O^*((k-1+\varepsilon)^r)$.*

General CSP We will use a k-SAT oracle (just presented) and clever reduction to reduce general CSP to k-SAT. Thus prooving the following:

Theorem 2. *There exists a deterministic algorithm having running time $O^*((d/2)^n)$ which takes any $(d, \leq k)$-CSP F over n variables and produces $l = O^*((d/2)^n)$ Boolean k-CNF formulas $\{\varphi_i\}_{1 \leq i \leq l}$ such that F is satisfiable if and only if there exists i such that φ_i is satisfiable.*

Corollary 1. *For every $\varepsilon > 0$, there is a deterministic algorithm solving $(d, \leq k)$-CSP in time $O * ((\frac{d(k-1)}{k} + \varepsilon)^n)$.*