Randomized local search for 3-CNF-SAT

Holger Dell

January 18, 2021

Introduction

In the context of the P vs. NP problem, an algorithm is said to be *efficient* if its worst-case running time is bounded by a polynomial in the input size. In the following few lectures, we will focus on a more subtle and more modest question:

> Is there an algorithm that's faster than exhaustive search?

For example, the satisfiability problem for Boolean circuits with n input gates and m wires can be solved in time $2^n \cdot \mathrm{poly}(m)$. However, it turns out that d-CNF SAT has significantly faster algorithms. Recall that d-CNF formulas F are formulas with $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ where each C_i is a clause of the form $C_i = (\ell_{i1} \vee \cdots \vee \ell_{id})$. We will prove that there is a randomized algorithm for d-CNF-SAT that runs in time $(2 - 2/d)^n \cdot \mathrm{poly}(m)$. For ease of notation, we focus on the case $d = 3$.

Randomized local search for 3-SAT

We now turn to a randomized algorithm by Schöning (2002), which solves 3-CNF-SAT in time $(4/3)^n \cdot \mathrm{poly}(m)$. In order to achieve this significant improvement over exhaustive search, we use some kind of gradient descent to find the satisfying assignment. Our measure for how "close" our current candidate is to a solution uses the *Hamming distance* d_H between two strings $x, y \in \{0,1\}^n$, which is defined as follows:

$$d_H(x,y) = \#\big\{i \in \{1,\dots,n\} : x_i \neq y_i\big\}.$$

The local search algorithm receives as input a 3-CNF formula F and a current candidate solution $a \in \{0,1\}^n$. The algorithm will only ever declare that the formula F is satisfiable if it successfully finds a satisfying assignment; therefore we can focus on the case in which there exists a satisfying assignment $a^* \in \{0,1\}^n$ for the formula F, that is, we have $F(a^*) = 1$.

The main idea for the local search algorithm is to iteratively flip individual bits of a in order to find the global minimum of the cost function $a \mapsto d_H(a, a^*)$. This minimum is achieved when $d_H(a, a^*) = 0$, which means that $a = a^*$. There are two problems with this approach:

1. The cost function $d_H(a, a^*)$ is far from convex, and so the algorithm may get stuck in a local minimum.
2. We don't know at all how $a \mapsto d_H(a, a^*)$ could possibly be computed efficiently since the algorithm doesn't know a^* in advance.

To overcome the first problem, we will restart the local search several times, and each time we will start from a uniformly chosen assignment $a \in \{0,1\}^n$. For the second problem, we make the following critical observation.

Observation. Let $F = C_1 \wedge \cdots \wedge C_m$ be a 3-CNF formula, let a^* be a satisfying assignment of F, and let a be an assignment with $F(a) = 0$. Then there exists an unsatisfied clause C_i, that is, we have $C_i(a) = 0$. Since $C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$, this implies $\ell_{i1} = \ell_{i2} = \ell_{i3} = 0$ in the assignment a. On the other hand, since a^* is a satisfying assignment, we have $\ell_{i1} = 1$ or $\ell_{i2} = 1$ or $\ell_{i3} = 1$ for a^*. This implies that we can flip in a the value of one of the literals ℓ_{i1}, ℓ_{i2}, or ℓ_{i3} in order to decrease the Hamming distance between a and a^*! We don't know *which* literal we should flip, but one of the three choices will bring us closer to a^*.

With this observation in mind, we can now formulate the main idea of the randomized local search algorithm: We simply select one of the three literals uniformly at random and flip its value in a. Before we think about how likely it is that we will actually find a solution, and how long it will take, let us formally state the local search algorithm `LocalSearch`(F, a):

- Loop at most $3n$ times:
  - If $F(a) = 1$, the algorithm found a satisfying assignment and outputs 1.
  - Otherwise, find a clause $C_i = (\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$ that is not satisfied by a.
  - Select a literal in that clause by choosing $j \in \{1, 2, 3\}$ uniformly at random.
  - Flip in a the bit corresponding to ℓ_{ij}.
- Give up and abort this round of `LocalSearch`, outputting 0.

As you can see, we loop at most $3n$ times; the rationale for this is that we assume that we are stuck in a local minimum if we haven't found a solution within $3n$ steps.

Analysis of the local search

Due to the cut-off after iteration $3n$, the running time of `LocalSearch` is polynomial. However, since 3-CNF-SAT is NP-complete, we don't

think that 3-CNF-SAT has a polynomial-time algorithm, not even one that is randomized and has a large success probability. Therefore, we expect the probability p that `LocalSearch` succeeds to be exponentially small. In fact, we will prove that it is roughly $(3/4)^n$.

Lemma 1. *Let F be a fixed 3-CNF formula with a satisfying assignment a^*. We choose the initial a uniformly at random from $\{0,1\}^n$. Then we have:*

$$p := \Pr(\mathsf{LocalSearch}(F, a) = 1) \geq (3/4)^n / \mathit{poly}(n)\,.$$

Proof. Let $X_t = d_H(a_t, a^*)$ where a_t is the value of the variable a after iteration t. Due to the initial random choice of a_0 and the random choices throughout the algorithm, a_t and X_t are random variables. Note that the expectation of X_0 is $E[X_0] = n/2$ and that X_0 is binomially distributed, that is, we have

$$\Pr(X_0 = j) = 2^{-n}\binom{n}{j}\,.$$

By the observation that we made above, we have

$$\Pr(X_{t+1} = X_t - 1 \mid X_t > 0) \geq \frac{1}{3}$$

and

$$\Pr(X_{t+1} = X_t + 1 \mid X_t > 0) \leq \frac{2}{3}\,.$$

Note that, because the latter inequality is an upper bound on the probability, it holds regardless of the conditioning. For simplicity, we analyze the random process where the two inequalities hold with equality:

$$\Pr(X_{t+1} = X_t - 1 \mid X_t > 0) = \frac{1}{3}$$

and

$$\Pr(X_{t+1} = X_t + 1 \mid X_t > 0) = \frac{2}{3}\,.$$

Exercise. Explain why we can do this simplification and where we are using it below.

We want to lower bound the probability

$$p = \sum_{t=0}^{3n} \Pr(X_t = 0 \mid \forall k < t . X_k > 0)\,.$$

For this, we condition on $X_0 = j$ in each term, which yields

$$
\begin{aligned}
p &= \sum_{t=0}^{3n}\sum_{j=0}^{n} \Pr(X_0 = j)\cdot \Pr(X_t = 0 \mid X_0 = j \wedge \forall k < t. X_k > 0) \\
&= \sum_{t=0}^{3n}\sum_{j=0}^{n} 2^{-n}\binom{n}{j}\cdot q(t,j) \\
&= \sum_{j=0}^{n} 2^{-n}\binom{n}{j}\cdot \sum_{t=0}^{3n} q(t,j)
\end{aligned}
$$

Here we defined $q(t,j)$ as the probability that, for a fixed a with $d_H(a,a^*) = j$, the algorithm `LocalSearch` finds the solution a^* after t iterations and not any earlier. As an example, consider $q(j,j)$: The only way in which `LocalSearch` could find the solution after j steps is to decrement the distance from a to a^* in every iteration; this happens with probability $q(j,j) = 3^{-j}$ as the probability of decrementing is exactly $1/3$ in each iteration independently. For the general case, note that the distance from a to a^* must be decremented in $j + (t-j)/2$ iterations and incremented in $(t-j)/2$ iterations, and it must always be bigger than 0. Thus we have $q(t,j) = (\frac{1}{3})^{(t+j)/2}(\frac{2}{3})^{(t-j)/2}\cdot Q(t,j)$, where $Q(t,j)$ is the number of such sequences.

Exercise. Let $t, j \in \mathbf{N}$. Prove that the number $Q(t,j)$ of sequences $s \in \{-1,1\}^t$ such that $j + \sum_{i=1}^{k} s_i > 0$ for all $k < t$ and $j + \sum_{i=1}^{t} s_i = 0$ hold is $\binom{t}{(t-j)/2}\cdot \frac{j}{t}$.

Exercise. Is the corresponding section on wikipedia wrong? If so, fix it.

We now continue our estimation of p by first proving $\sum_t q(t,j) \sim 2^{-j}$, where $\sim$ indicates equality up to factor that is polynomial in t:

$$
\begin{aligned}
\sum_{t=0}^{3n} q(t,j) &= \sum_{t=j}^{3n} q(t,j) \\
&\sim \sum_{t\geq j} \binom{t}{(t-j)/2} \frac{2^{(t-j)/2}}{3^{(t+j)/2+(t-j)/2}} \\
&= \sum_{t\geq j} \binom{t}{\alpha t}\cdot 2^{\alpha t}\cdot 3^{-t} \\
&\sim \max_{t\geq j} \binom{t}{\alpha t}\cdot 2^{\alpha t}\cdot 3^{-t} \\
&\sim \max_{t\geq j} 2^{H(\alpha)t}\cdot 2^{\alpha t}\cdot 3^{-t}\,.
\end{aligned}
$$

Here we wrote $\alpha = \frac{1}{2} - \frac{j}{2t}$ and used the following important fact about the binomial coefficient:

Exercise. Use Stirling's approximation of $n!$ to prove that $\binom{t}{\alpha t} \sim 2^{H(\alpha)t}$ holds for all $\alpha \in (0, \frac{1}{2})$ and $t \in \mathbb{N}$, where $H(\alpha)$ is the binary entropy of α:

$$H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)\,.$$

By setting $\frac{\partial}{\partial t}(H(\alpha)t + \alpha t - t) = 0$, we find the value of t for which the maximum is achieved, and we find that the value of the maximum is $\sim 2^{-j}$

Exercise. Perform this calculation to find the maximum.

Finally, for our estimate for p, we get

$$\begin{aligned} p &= 2^{-n} \cdot \sum_{j=0}^{n} \binom{n}{j} \sum_{t} q(t,j) \\ &\sim 2^{-n} \cdot \sum_{j=0}^{n} \binom{n}{j} 2^{-j} \cdot 1^{n-j} \\ &= 2^{-n} \cdot (\frac{1}{2} + 1)^n \\ &= (3/4)^n\,. \end{aligned}$$

This finishes the proof of the lemma. □

Probability amplification

Let $p = (3/4)^n$ be the success probability of `LocalSearch`. By repeating `LocalSearch` $20/p$ times, we bring its one-sided error probability down to at most e^{-20}.

Theorem 1. *There is a randomized algorithm for 3-CNF-SAT that has running time* $(4/3)^n \cdot poly(n)$ *and success probability* $1 - e^{-20}$.