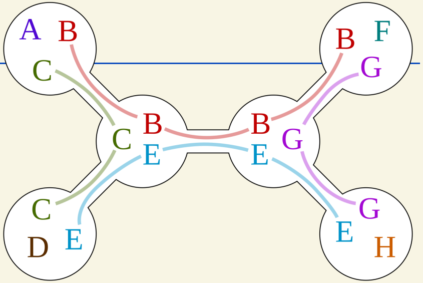
Randomized Methods II: Walk-SAT

DM898: Parameterized Algorithms

Lars Rohwedder



Today's lecture

- Improvement over $O(2^n)$ for k-SAT via a random walk

Satisfiability problem

k-SAT

A k-SAT formula consists of n Boolean variables $x_1, \ldots, x_n$ and a logical AND over m clauses, each forming a logical OR of at most k literals. A literal is a variable x_i or a negated variable $\neg x_i$.

Input: k-SAT formula

Output: YES, if there exists a satisfying assignment; NO, otherwise

Naive algorithm: $O(2^n)$ time. **Can we do better?**

Satisfiability problem

k-SAT

A k-SAT formula consists of n Boolean variables $x_1, \dots, x_n$ and a logical AND over m clauses, each forming a logical OR of at most k literals. A literal is a variable x_i or a negated variable $\neg x_i$.

Input: k-SAT formula

Output: YES, if there exists a satisfying assignment; NO, otherwise

Naive algorithm: $O(2^n)$ time. **Can we do better?**

We have seen in the exercises a $O(1.74^n)$ algorithm for 3-SAT via branching. Idea:

- For any satisfying assignment, $x_1 = x_2 = \cdots = x_n = \text{false}$ or $x_1 = x_2 = \cdots = x_n = \text{true}$ differs in at most $n/2$ variables. Branch on which one is the case and select this as initial solution
- As long as the assignment is not satisfying (and the recursion depth is $\leq n/2$), select a clause C that is not satisfies and branch on a variable in C to flip
- Running time: $O(k^{n/2})$

Satisfiability problem

k-SAT

A k-SAT formula consists of n Boolean variables $x_1, \ldots, x_n$ and a logical AND over m clauses, each forming a logical OR of at most k literals. A literal is a variable x_i or a negated variable $\neg x_i$.

Input: k-SAT formula

Output: YES, if there exists a satisfying assignment; NO, otherwise

Naive algorithm: $O(2^n)$ time. **Can we do better?**

We have seen in the exercises a $O(1.74^n)$ algorithm for 3-SAT via branching. Idea:

- For any satisfying assignment, $x_1 = x_2 = \cdots = x_n = \text{false}$ or $x_1 = x_2 = \cdots = x_n = \text{true}$ differs in at most $n/2$ variables. Branch on which one is the case and select this as initial solution
- As long as the assignment is not satisfying (and the recursion depth is $\leq n/2$), select a clause C that is not satisfies and branch on a variable in C to flip
- Running time: $O(k^{n/2})$ $\leftarrow$ **better than $O(2^n)$ iff $k < 4$**

What about $k \geq 4$?

Walk-SAT

Algorithm

- start with a variable assignment $x_1, x_2, \ldots, x_n$ sampled independently and uniformly at random
- repeat N times (or until all clauses are satisfied):
  choose arbitrary clause C that is violated, sample a variable from C uniformly at random and flip it

Here, cutoff N needs to be decided. The procedure is repeated (each time with a new random starting solution) to boost its probability of success

- Very similar to local search heuristics used on various problems in practice
- Walk-SAT is a serious approach for solving SAT in practice, see e.g. `https://www.cs.virginia.edu/~rmw7my/walksat/index.html`
- Schöning analyzed this algorithm for k-SAT and proved that with $N = 3n$ and $O((2 - 2/k)^n)$ iterations the algorithm succeeds with probability 0.99, see[1]

[1] A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. Uwe Schöning. 1999

A first analysis

Success bound of one iteration

- Let x^* be a satisfying assignment and let $x^{(0)}, x^{(1)}, x^{(2)} \dots$ be the assignments in throughout the iterations.
- Let $d(x^*, x^{(j)})$ be the Hamming distance (number of different values) of the two assignments.

If clause C is not satisfied by $x^{(j)}$, then one of the k variables in C must be set differently in x^* and $x^{(j)}$. Hence, we have probability $\geq 1/k$ that $d(x^*, x^{(j+1)}) = d(x^*, x^{(j)}) - 1$.

Success bound of one iteration

- Let x^* be a satisfying assignment and let $x^{(0)}, x^{(1)}, x^{(2)} \dots$ be the assignments in throughout the iterations.
- Let $d(x^*, x^{(j)})$ be the Hamming distance (number of different values) of the two assignments.

If clause C is not satisfied by $x^{(j)}$, then one of the k variables in C must be set differently in x^* and $x^{(j)}$. Hence, we have probability $\geq 1/k$ that $d(x^*, x^{(j+1)}) = d(x^*, x^{(j)}) - 1$.

If this happens for each of the first $d(x^*, x^{(0)}) \leq N$ iterations, then the algorithm succeeds. Thus, success probability $\geq$

$$\left(\frac{1}{k}\right)^{d(x^*,x^{(0)})}$$

. . . similar to the running time of the branching algorithm. What can we say about $d(x^*, x^{(0)})$?

Distribution of initial assignment

- let S be the set of indices where x^* and $x^{(0)}$ differ
- for each variable $i \in \{1, 2, \ldots, n\}$ we have $i \in S$ and $i \notin S$ with equal probability
- Hence, for every $T \subseteq \{1, 2, \ldots, n\}$ we have $\mathbb{P}[S = T] = 1/2^n$

It follows that for each $i \in \{0, 1, 2, \ldots, n\}$

$$\mathbb{P}[d(x^*, x^{(0)}) = i] = \sum_{T \subseteq \{1,2,\ldots,n\} : |T| = i} \mathbb{P}[S = T] = 2^{-n} \binom{n}{i}$$

Distribution of initial assignment

- let S be the set of indices where x^* and $x^{(0)}$ differ
- for each variable $i \in \{1, 2, \dots, n\}$ we have $i \in S$ and $i \notin S$ with equal probability
- Hence, for every $T \subseteq \{1, 2, \dots, n\}$ we have $\mathbb{P}[S = T] = 1/2^n$

It follows that for each $i \in \{0, 1, 2, \dots, n\}$

$$\mathbb{P}[d(x^*, x^{(0)}) = i] = \sum_{T \subseteq \{1,2,\dots,n\} : |T| = i} \mathbb{P}[S = T] = 2^{-n} \binom{n}{i}$$

With previous bound, success probability of WalkSAT is $\geq$

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

Binomial Theorem

For every $x, y \in \mathbb{R}$

$$\sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i} = (x + y)^n$$

Distribution of initial assignment

- let S be the set of indices where x^* and $x^{(0)}$ differ
- for each variable $i \in \{1, 2, \ldots, n\}$ we have $i \in S$ and $i \notin S$ with equal probability
- Hence, for every $T \subseteq \{1, 2, \ldots, n\}$ we have $\mathbb{P}[S = T] = 1/2^n$

It follows that for each $i \in \{0, 1, 2, \ldots, n\}$

$$\mathbb{P}[d(x^*, x^{(0)}) = i] = \sum_{T \subseteq \{1,2,\ldots,n\} : |T| = i} \mathbb{P}[S = T] = 2^{-n} \binom{n}{i}$$

With previous bound, success probability of WalkSAT is $\geq$

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

Binomial Theorem

For every $x, y \in \mathbb{R}$

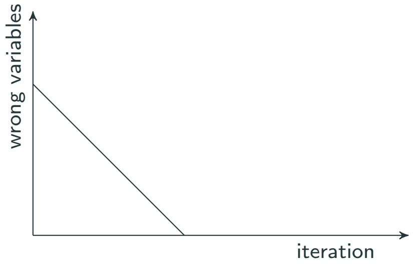
$$\sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i} = (x + y)^n$$

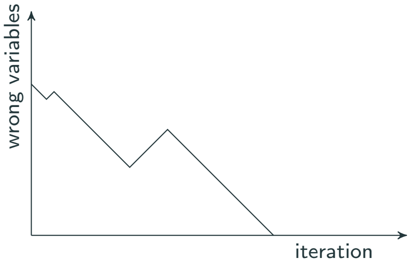
Hence, after $O((2 - 2/(k+1))^n)$ iterations, the success probability is 0.99

Careful analysis for 3-SAT

Random walk

We considered only the direct walk from i wrong variables to 0, but moving slightly less efficiently to the correct assignment would also be fine.





Goal: bound probability that a random walk arrives at 0 (after not too many steps), starting at $i \in \mathbb{N}$ and moving -1 w.p. $\geq 1/3$ and $+1$ w.p. $\leq 1 - 1/3$ each step.

Analyzing the random walk

Probability that a random walk arrives at 0 (after at most $N = 3n$ steps), starting at i and moving $+1$ w.p. $\leq 1 - 1/3$ and -1 w.p. $\geq 1/3$ each step.

Analyzing the random walk

Probability that a random walk arrives at 0 (after at most $N = 3n$ steps), starting at i and moving $+1$ w.p. $\leq 1 - 1/3$ and -1 w.p. $\geq 1/3$ each step.

$\leq$ probability that a random walk arrives at 0 (after at most $3n$ steps), starting at i and moving $+1$ w.p. $= 1 - 1/3$ and -1 w.p. $= 1/3$ each step.

Analyzing the random walk

Probability that a random walk arrives at 0 (after at most $N = 3n$ steps), starting at i and moving $+1$ w.p. $\leq 1 - 1/3$ and -1 w.p. $\geq 1/3$ each step.

$\leq$ probability that a random walk arrives at 0 (after at most $3n$ steps), starting at i and moving $+1$ w.p. $= 1 - 1/3$ and -1 w.p. $= 1/3$ each step.

$\leq$ probability that a random walk arrives at 0 **after exactly $3i$ steps**, starting at i and moving $+1$ w.p. $= 1 - 1/3$ and -1 w.p. $= 1/3$ each step.

Analyzing the random walk

Probability that a random walk arrives at 0 (after at most $N = 3n$ steps), starting at i and moving $+1$ w.p. $\leq 1 - 1/3$ and -1 w.p. $\geq 1/3$ each step.

$\leq$ probability that a random walk arrives at 0 (after at most $3n$ steps), starting at i and moving $+1$ w.p. $= 1 - 1/3$ and -1 w.p. $= 1/3$ each step.

$\leq$ probability that a random walk arrives at 0 **after exactly $3i$ steps**, starting at i and moving $+1$ w.p. $= 1 - 1/3$ and -1 w.p. $= 1/3$ each step.

We estimate the probability of going down $2i$ times and up i times and simplify:

$$\binom{3i}{i}\left(\frac{1}{3}\right)^{2i}\left(\frac{2}{3}\right)^{i}$$

$$\geq \Omega\left(\sqrt{\frac{3i}{2i^2}} \cdot \frac{(3i)^{3i}}{i^i \cdot (2i)^{2i}} \cdot \left(\frac{1}{3}\right)^{2i}\left(\frac{2}{3}\right)^{i}\right)$$

$$\geq \Omega\left(\sqrt{\frac{1}{i}} \cdot \frac{3^{3i}}{2^{2i}}\left(\frac{1}{3}\right)^{2i}\left(\frac{2}{3}\right)^{i}\right)$$

$$\geq \Omega\left(\sqrt{\frac{1}{i}}\left(\frac{1}{2}\right)^{2i} 2^i\right) \geq \Omega\left(\sqrt{\frac{1}{i}}\left(\frac{1}{2}\right)^{i}\right)$$

Stirling's approximation for binomial coefficient

$$\binom{n}{k} = \Theta\left(\sqrt{\frac{n}{k(n-k)}} \cdot \frac{n^n}{k^k(n-k)^{n-k}}\right)$$

Analyzing the random walk

Probability that a random walk arrives at 0 (after at most $N = 3n$ steps), starting at i and moving $+1$ w.p. $\leq 1 - 1/3$ and -1 w.p. $\geq 1/3$ each step.

$\leq$ probability that a random walk arrives at 0 (after at most $3n$ steps), starting at i and moving $+1$ w.p. $= 1 - 1/3$ and -1 w.p. $= 1/3$ each step.

$\leq$ probability that a random walk arrives at 0 **after exactly $3i$ steps**, starting at i and moving $+1$ w.p. $= 1 - 1/3$ and -1 w.p. $= 1/3$ each step.

We estimate the probability of going down $2i$ times and up i times and simplify:

$$\binom{3i}{i}\left(\frac{1}{3}\right)^{2i}\left(\frac{2}{3}\right)^{i}$$

$$\geq \Omega\left(\sqrt{\frac{3i}{2i^2}} \cdot \frac{(3i)^{3i}}{i^i \cdot (2i)^{2i}} \cdot \left(\frac{1}{3}\right)^{2i}\left(\frac{2}{3}\right)^{i}\right)$$

$$\geq \Omega\left(\sqrt{\frac{1}{i}} \cdot \frac{3^{3i}}{2^{2i}}\left(\frac{1}{3}\right)^{2i}\left(\frac{2}{3}\right)^{i}\right)$$

$$\geq \Omega\left(\sqrt{\frac{1}{i}}\left(\frac{1}{2}\right)^{2i} 2^i\right) \geq \Omega\left(\sqrt{\frac{1}{i}}\left(\frac{1}{2}\right)^{i}\right)$$

Stirling's approximation for binomial coefficient

$$\binom{n}{k} = \Theta\left(\sqrt{\frac{n}{k(n-k)}} \cdot \frac{n^n}{k^k(n-k)^{n-k}}\right)$$

Taking the expectation over all initial number of mistakes i, we obtain **probability of success** $\geq$

$$\Omega\left(2^{-n}\sum_{i=1}^{n}\binom{n}{i}\sqrt{\frac{1}{i}}\left(\frac{1}{2}\right)^{i}\right) \geq \Omega\left(2^{-n}\frac{1}{\sqrt{n}}\left(1+\frac{1}{2}\right)^{n}\right)$$

$$\geq \Omega\left(\frac{1}{\sqrt{n}}\left(\frac{3}{4}\right)^{n}\right)$$

Wrapping up

One iteration of the algorithm takes polynomial time and has success probability $\geq \Omega(1/\sqrt{n} \cdot (3/4)^n)$.

By repeating the algorithm $O(\sqrt{n} \cdot (4/3)^n)$ times we obtain an algorithm for 3-SAT with constant success probability and running time

$$\left(\frac{4}{3}\right)^n \cdot n^{O(1)}$$

Based on similar (but technically more involved) ideas, the analysis can be generalized to any $k \in \mathbb{N}$, leading to an algorithm for k-SAT with running time

$$\left(2 - \frac{2}{k}\right)^n \cdot n^{O(1)}$$

Strong Exponential Time Hypothesis

The previous algorithm's running time is roughly $O(2^n)$ for $k \to \infty$. This motivates the following hypothesis:

Strong Exponential Time Hypothesis (SETH)

For every $\epsilon > 0$ there exists $k \in \mathbb{N}$ such that k-SAT does not have an $O(2^{(1-\epsilon)n})$ time algorithm

SETH is even more controversial than ETH, since it is an even stronger hypothesis (see next slide), and of course also not proven.

SETH has surprising implications for polynomial time solvable problems. For example, it can be shown that SETH implies that there is no subquadratic (i.e., $O(n^{1.999})$-time) algorithm for Longest-Common-Subsequence.

Such results form the area of **fine-grained complexity**, for which we will see a few other examples with proofs soon.

SETH implies ETH

Exponential Time Hypothesis (ETH)

There exists some constant $\epsilon > 0$ such that there is no $O(2^{\epsilon n})$ time algorithm for 3-SAT

Strong Exponential Time Hypothesis (SETH)

For every $\epsilon > 0$ there exists $k \in \mathbb{N}$ such that k-SAT does not have an $O(2^{(1-\epsilon)n})$ time algorithm

To prove that SETH implies ETH, we need the general version of the Sparsification Lemma. We do not give a proof of this here. It is similar to the proof for 3-SAT we have seen (but more involved)

Sparsification Lemma (general form)

Let $k \in \mathbb{N}$. For every $\epsilon > 0$ there is some $\epsilon' > 0$ such that if there exists an $O(2^{\epsilon'(n+m)})$ time algorithm for k-SAT, then there also exists an $O(2^{\epsilon n})$ time algorithm for it

SETH implies ETH

Exponential Time Hypothesis (ETH)

There exists some constant $\epsilon > 0$ such that there is no $O(2^{\epsilon n})$ time algorithm for 3-SAT

Strong Exponential Time Hypothesis (SETH)

For every $\epsilon > 0$ there exists $k \in \mathbb{N}$ such that k-SAT does not have an $O(2^{(1-\epsilon)n})$ time algorithm

To prove that SETH implies ETH, we need the general version of the Sparsification Lemma. We do not give a proof of this here. It is similar to the proof for 3-SAT we have seen (but more involved)

Sparsification Lemma (general form)

Let $k \in \mathbb{N}$. For every $\epsilon > 0$ there is some $\epsilon' > 0$ such that if there exists an $O(2^{\epsilon'(n+m)})$ time algorithm for k-SAT, then there also exists an $O(2^{\epsilon n})$ time algorithm for it

Proof that SETH implies ETH.

- Assume for every $\epsilon > 0$ there is an algorithm for 3-SAT with running time $O(2^{\epsilon n})$
- Consider a k-SAT formula with n variables and m clauses. Replace every clause $l_1 \vee l_2 \vee \cdots \vee l_h$, $3 < h \leq k$, by 2- and 3-clauses $(l_1 \vee y_1), (\neg y_1 \vee l_2 \vee y_2), (\neg y_2 \vee l_3 \vee y_3), \ldots, (\neg y_{h-1} \vee l_h)$ where $y_1, \ldots, y_{h-1}$ are new variables
- The resulting 3-SAT formular is equivalent to the k-SAT formula and has $n' \leq n + m(k-1)$ variables
- Let $\epsilon > 0$. Then by the previous assumptions and with $\epsilon' := \epsilon/(k-1)$, we can solve the instance above in time $O(2^{\epsilon' n'}) \leq O(2^{\epsilon(n+m)})$, which by the Sparsification Lemma leads to e.g. an $O(2^{n/2})$ time algorithm for k-SAT for any $k = O(1)$, contradicting SETH