Computationally Hard Problems

Design of Randomized Algorithms: MAX-SAT

Carsten Witt

Institut for Matematik og Computer Science
Danmarks Tekniske Universitet

Fall 2020

The Problem

Problem [MaximumSatisfiability]
Input: A set of clauses over n boolean variables (and a natural number k for the decision version).
Output for the decision version: YES if there is an assignment of truth values to the variables which satisfies at least k clauses and NO otherwise.
Output for the optimizing version: An assignment of truth values to the variables which satisfies a maximal number of clauses.

A Simple Randomized Solution

Let $C = \{c_1, c_2, \ldots, c_m\}$ be a set of clauses over the Boolean variables $x_1, x_2, \ldots, x_n$.

We want to show that there is an assignment of truth values to the x_i which satisfies at least $m/2$ clauses.

We assign truth values to the variables at random.

for $i = 1, \ldots, n$ **do**
  if $(coin = 0)$ **then**
    $x_i \leftarrow$ **false**
  else
    $x_i \leftarrow$ **true**
  end if
end for

A Simple Randomized Solution

Let $c_j = x_1 \vee \cdots \vee x_{k_j}$ a clause.

The prob. that c_j is **not** satisfied by the random assignment is $(1/2)^{k_j}$. Why?

For every literal of c_j the probability to be not satisfied is $(1/2)$.

The probability that all literals of c_j are not satisfied is $(1/2)^{k_j}$.

The probability that some literal of c_j is satisfied is $1 - (1/2)^{k_j}$.

The probability s_j that c_j is satisfied is

$$s_j = 1 - (1/2)^{k_j}$$

Observation: Longer clauses are more easily satisfied.
Observation: For every clause c_j

$$s_j = 1 - (1/2)^{k_j} \geq 1 - (1/2) \geq 1/2.$$

Let N be the number of clauses satisfied.

N is a random variable.

The expected value is

$$\boldsymbol{E}[N] = \sum_{j=1}^{m} s_j .$$

Using the observation this gives

$$\boldsymbol{E}[N] \geq \sum_{j=1}^{m} \frac{1}{2} = \frac{m}{2}.$$

Since $\boldsymbol{E}[N] \geq m/2$, there must be at least one assignment that satisfies at least $m/2$ clauses. Otherwise the expectation would be smaller (Fact A.1 in lecture notes, *probabilistic method*).

Approximation Ratio

Let $A(C)$ be the number of clauses from $C = \{c_1, c_2, \ldots, c_m\}$ satisfied by algorithm A.

Let $\mathrm{OPT}(C)$ be the maximal number of clauses that can be satisfied.

Recall that if A is randomized then $R_A(C)$ is defined to be the *expected approximation ratio*:

$$R_A(C) = \frac{\mathrm{OPT(C)}}{\boldsymbol{E}\,[\,A(C)\,]}\ .$$

The last algorithm has an expected approximation ratio of 2.

Outline

The aim is to improve the ratio (to $4/3$).

The above algorithm has a bad performance on short clauses.

It is good on long clauses.

We describe a different algorithm based on formulating $\textsc{MaxSat}$ as linear program.

We run both algorithms.

Finally we select the better solution.

Integer Programming

Problem [INTEGERPROGRAMMING]
Input: Parameters $c_1, \dots, c_m \in \mathbb{Z}$, $a_{j1}, \dots, a_{jm}, b_j \in \mathbb{Z}$, $j = 1, \dots, k$.
Output for the optimizing version: Values $x_1, \dots, x_m \in \mathbb{Z}$ such that

- The target function $c_1 x_1 + \cdots + c_m x_m$ is maximized.
- The constraints are met $a_{j1} x_1 + \cdots + a_{jm} x_m \leq b_j, \quad j = 1, \dots, k$

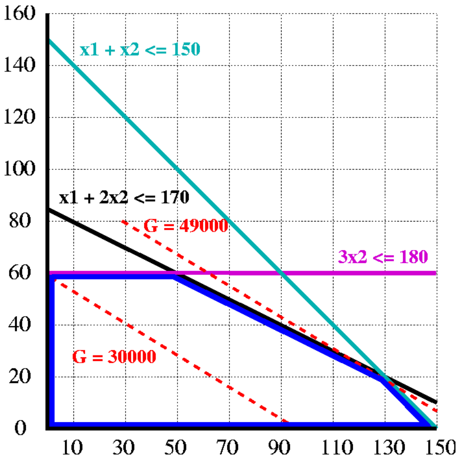
The problem is $\mathcal{NP}$-hard. If the numbers x_i are allowed to be reals then the problem is efficiently solvable.

Integer Programming, Example

$$\begin{aligned} 300x_1 + 500x_2 &= \textbf{max!} \\ x_1 + 2x_2 &\leq 170 \\ x_1 + x_2 &\leq 150 \\ 3x_2 &\leq 180 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{Z} \end{aligned}$$

The maximal value 49000 of the target function is achieved by setting $x_1 = 130$, $x_2 = 20$.

This example has the same optimal solution if we allow $x_1, x_2 \in \mathbb{R}$: that is not typical.



From Clauses to Constraints

Let $c_1, \ldots, c_m$ be clauses over $x_1, \ldots, x_n$.

The linear program uses variables $z_1, \ldots, z_m$ and $y_1, \ldots, y_n$. (Since $x_1, \ldots, x_n$ is taken.)

For the integer program we have $z_j, y_i \in \{0, 1\}$.

Interpretation: $z_j = 1$ iff clause c_j is satisfied.

Interpretation: $y_i = 1$ iff boolean variable $x_i = \mathsf{true}$.

Target function: $z_1 + \cdots + z_m$.

Maximizing the target function means satisfying as many clauses as possible.

From Clauses to Constraints

Let $c_1, \ldots, c_m$ be clauses over $x_1, \ldots, x_n$.

Consider clause $c_j = x_a \vee \overline{x_b} \vee x_c$

This is modelled by the following constraint

$$y_a + (1 - y_b) + y_c \geq z_j$$

Interpretation

- In order to maximize the target function, z_j should be one.
- This can only be obstructed by the left-hand side of the constraint being 0.
- Then $y_a = 0$, $y_b = 1$, and $y_c = 0$.
- Then c_j is not satisfied.

Example

$$
\begin{aligned}
X &= \{x_1, x_2\} \\
c_1 &= x_1 \vee \overline{x_2} \\
c_2 &= \overline{x_1} \vee \overline{x_2} \\
c_3 &= x_2
\end{aligned}
$$

$$
\begin{aligned}
\textbf{max!} \quad & z_1 + z_2 + z_3 \\
y_1 + (1 - y_2) &\geq z_1 \\
(1 - y_1) + (1 - y_2) &\geq z_2 \\
y_2 &\geq z_3 \\
y_i &\in \{0, 1\}, i = 1, 2 \\
z_j &\in \{0, 1\}, j = 1, 2, 3
\end{aligned}
$$

Relaxing the Program

Integer programs are hard to solve.

We *relax* it to a linear program.

The conditions $y_i \in \{0, 1\}$ are replaced by

$$y_i \geq 0 \text{ and } y_i \leq 1$$

and accordingly for z_j.

This kind of programs is efficiently solvable.

But we sacrificed the interpretation of y_i.

If the solution is $y_i = 0.6$ what does this mean,
"x_i is more true than false"?

Example, Relaxed Program

$$
\begin{aligned}
X &= \{x_1, x_2\} \\
c_1 &= x_1 \vee \overline{x_2} \\
c_2 &= \overline{x_1} \vee \overline{x_2} \\
c_3 &= x_2
\end{aligned}
$$

$$
\begin{array}{rcll}
\textbf{max!} & & z_1 + z_2 + z_3 & \\
y_1 + (1 - y_2) & \geq & z_1 & \\
(1 - y_1) + (1 - y_2) & \geq & z_2 & \\
y_2 & \geq & z_3 & \\
y_i & \geq & 0 & i = 1, 2 \\
y_i & \leq & 1 & i = 1, 2 \\
z_i & \geq & 0 & i = 1, 2, 3 \\
z_i & \leq & 1 & i = 1, 2, 3
\end{array}
$$

Randomized Rounding

The solution $\hat{y}_1, \ldots, \hat{y}_n, \hat{z}_1, \ldots, \hat{z}_m$ of the relaxed program can be non-integer, e.g., $\hat{y}_i = 0.6$.

This can be interpreted as 60% true.

For a truth assignment we need (100%) true or false.

So, we round to an integer.

But not necessarily to the nearest one; that might not be optimal.

On the other hand, the value of $\hat{y}_i$ *might* indicate whether $x_i = \text{true}$ or $x_i = \text{false}$ is a better choice.

We use *randomized rounding*:

$$x_i = \begin{cases} \text{true}, & \text{with probability } \hat{y}_i; \\ \text{false}, & \text{with probability } 1 - \hat{y}_i. \end{cases}$$

Analysis of Randomized Rounding (1/3)

Overview

- Let c_j consist of k literals. Then the probability that c_j is satisfied by randomized rounding is at least
$$\left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot \hat{z}_j$$
- $1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - (1/e) \sim 0.632$.
- Let $N_{\max}$ be the maximum number of clauses that can be satisfied. Then randomized rounding will satisfy at least $\left(1 - \frac{1}{e}\right) \cdot N_{\max} \sim 0.632 N_{\max}$ clauses in expectation.

Analysis of Randomized Rounding (2/3)

- Let $c_j = x_1 \vee x_2 \vee \cdots \vee x_k$.
- Assume c_j is not satisfied.
- Then all x_i in c_j are set false by randomized rounding.
- This happens with probability $\prod_{i=1}^{k}(1-\hat{y}_i)$, hence c_j is satisfied with prob. $1-\prod_{i=1}^{k}(1-\hat{y}_i)$.
- From $\hat{y}_1 + \hat{y}_2 + \cdots + \hat{y}_k \geq \hat{z}_j$, we have $(1-\hat{y}_1) + (1-\hat{y}_2) + \cdots + (1-\hat{y}_k) \leq k - \hat{z}_j$.
- Since $\sqrt[n]{\prod_{i=1}^{n} a_i} \leq \frac{1}{n}\sum_{i=1}^{n} a_i$ for non-negative a_i, we have
$$\prod_{i=1}^{k}(1-\hat{y}_i) \leq \left(\frac{(1-\hat{y}_1) + \cdots + (1-\hat{y}_k)}{k}\right)^k \leq \left(\frac{k-\hat{z}_j}{k}\right)^k .$$

Analysis of Randomized Rounding (3/3)

- Hence: clause c_j satisfied with prob. at least $1-\left(\frac{k-\hat{z}_j}{k}\right)^k = 1-\left(1-\frac{\hat{z}_j}{k}\right)^k$.
- Helper lemma: For $x \in [0,1]$, $1-(1-\frac{x}{k})^k \geq \left(1-(1-\frac{1}{k})^k\right) \cdot x$.
- Hence, the prob. that c_j is satisfied is at least $1-\left(1-\frac{\hat{z}_j}{k}\right)^k \geq \left(1-(1-\frac{1}{k})^k\right) \cdot \hat{z}_j$.
- The expected number of satisfied clauses is

$$\sum_{j=1}^{m}\left(1-\left(1-\frac{1}{k}\right)^k\right) \cdot \hat{z}_j \geq \left(1-\frac{1}{e}\right) \cdot \sum_{j=1}^{m} \hat{z}_j \geq \left(1-\frac{1}{e}\right) \cdot N_{\max} \,.$$

Combining the Best

Algorithm MIX:

- Let $C = \{c_1, c_2, \ldots, c_m\}$ be a set of clauses over the Boolean variables $x_1, x_2, \ldots, x_n$.
- Compute a solution with the "first" algorithm (by the way, this is randomized rounding with all $\hat{y}_i$ set to $1/2$).
- Set up and solve a relaxed linear program. Compute a solution with randomized rounding using the solutions $\hat{y}_i$ of the LP.
- You have two solutions (i. e., truth assignments) now. Take the one that satisfies more clauses.

Performance of MIX (1/4)

Let $c_1, c_2, \ldots, c_m$ be clauses over $x_1, x_2, \ldots, x_n$. Let $N_{\max}$ be the maximum number of clauses that can be satisfied. Then the solution of algorithm MIX satisfies at least $(3/4) \cdot N_{\max}$ clauses in expectation.

In other words: The expected approximation ratio is no worse than $\frac{N_{\max}}{(3/4)N_{\max}} = 4/3$.

To prove this, compare for every k the probabilities of the approaches to satisfy a single clause of length k and look at the average.

Let $A_k := 1 - 2^{-k}$ the factor from the first, $B_k := 1 - \left(1 - \frac{1}{k}\right)^k$ the factor from the second approach. Let $C_k := (A_k + B_k)/2$.

Performance of MIX (2/4)

k	A_k	B_k	C_k
1	0.5	1.0	0.75
2	0.75	0.75	0.75
3	0.875	0.704	0.7895
4	0.938	0.684	0.811
5	0.969	0.672	0.8205

As one cannot blend the results of two algorithms (but has to take one or the other), this is not yet a proof.

Performance of MIX (3/4)

Let N_1 and N_2 be the **expected** number of clauses satisfied by the first and second algorithm, respectively. We could like to prove $\max\{N_1, N_2\} \geq (3/4)N_{\max}$. This follows from $(N_1 + N_2)/2 \geq (3/4)N_{\max}$ (or $N_1 + N_2 \geq (3/2)N_{\max}$), which we (almost) prove in the following.

One detail is left out and can be found in the lecture notes.

Performance of MIX (4/4)

We show $A_k + B_k \geq 3/2$ for all $k \geq 1$.

Recall: $A_k + B_k = (1 - 2^{-k}) + \left(1 - (1 - \frac{1}{k})^k\right)$

For $k = 1$ and $k = 2$, we have $A_k + B_k = (3/2)$.

For $k \geq 3$, we have $A_k = (1 - 2^{-k}) \geq 7/8$ and $B_k = \left(1 - (1 - \frac{1}{k})^k\right) \geq 1 - \frac{1}{e}$.

Now, $\dfrac{7}{8} + 1 - \dfrac{1}{e} = 1.507\ldots \geq \dfrac{3}{2}$.

Since this holds for all k, the expected approximation ratio follows. □

Computationally Hard Problems

Design of Randomized Algorithms: 3-SAT

Carsten Witt

Institut for Matematik og Computer Science
Danmarks Tekniske Universitet

Fall 2020

The Decision Problem

Reconsider the classical decision problem $3\text{-}\textsc{Sat}$.

Using brute force, we can solve the problem in time 2^n. We cannot hope for a polynomial-time algorithm.

Can we still do better?

Yes, a randomized algorithm with expected running time $\approx 1.34^n$ is available.

This can make a huge difference. For example, $2^{25} = 33{,}554{,}432$, while $1.34^{25} \approx 1{,}329$. Randomized approach is $25{,}000$ times faster.

First of All: Solve an Easy Problem

Problem [2-SAT]
Input: A set of clauses $C = \{c_1, \dots, c_m\}$ over n boolean variables $x_1, \dots, x_n$, where every clause contains exactly two literals.
Output: YES if there is a satisfying assignment, i. e., if there is an assignment

$$a\colon \{x_1, \dots, x_n\} \to \{0, 1\}$$

such that every clause c_j is satisfied, and NO otherwise.

Can be solved in polynomial (even linear time) using classical results from logic.

Our algorithm will solve $2\text{-}\textsc{Sat}$ in polynomial time. Important ideas of its analysis will already become clear here.

The Randomized Algorithm

Algorithm RandomizedSat; input: set of clauses, S, T

for $s = 1, \ldots, S$ **do**

  for $i = 1, \ldots, n$ (independently) **do**

    $x_i \leftarrow 1$ with probability $1/2$, otherwise $x_i \leftarrow 0$.

  end for

  for $t = 1, \ldots, T$ **do**

    If $x = (x_1, \ldots, x_n)$ satisfies all clauses, output $\boldsymbol{x}$. STOP.

    Uniformly select a clause c_j that is not satisfied by x.

    Choose uniformly a literal z from c_j. Say, $z = x_i$ or $z = \overline{x_i}$.

    $x_i \leftarrow 1 - x_i$.

  end for

end for

Output: "There is probably no satisfying assignment."

How the Algorithm Works

Starts from a completely random assignment.

Flips setting of a variable randomly chosen from a unsatisfied clause.

E. g.: Unsatisfied clause $x_1 \vee \overline{x_5}$ chosen: Change assignment of x_1 from 0 to 1, or assignment of x_5 from 1 to 0 (each with prob. $1/2$).

Repeats this T times (if no satisfying assignment found so far).

Restarts everything from a fresh uniform assignment at most S times.

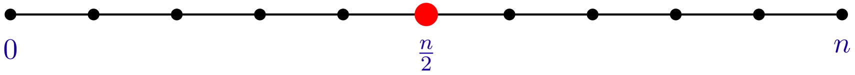
Altogether, a very simple algorithm with running time $O(ST(m+n))$. Why and how should we get polynomial time for 2-SAT?

Theorem: If given a satisfiable $2\text{-}\textsc{Sat}$ instance, then RandomizedSat with $T = \infty$ and $S = 1$ outputs a satisfying assignment after expected number $\leq n^2$ iterations of the inner loop.

Proof Technique: Fair Random Walk

Scenario "Fair Random Walk"

- Initially, player A and B both have $\frac{n}{2}$ DKK
- Repeat: flip a coin
- If heads: A pays 1 DKK to B, tails: other way round
- Until one of the players is ruined.



How long does the game take in expectation?

Theorem:

Fair random walk on $\{0, \ldots, n\}$ takes in expectation $\leq n^2$ steps, even if restarted until 0 is reached.

Proof of the Theorem

Assume satisfying assignment $x^* = (x_1^*, \ldots, x_n^*)$ (e. g. all variables 1) exists. Let $x = (x_1, \ldots, x_n)$ be current assignment, assume x not satisfying.

Consider $d(x, x^*) := \sum_{i=1}^n |x_i - x_i^*|$ ("distance"), number of variables differently assigned in x and x^*. Obviously, $0 \leq d(x, x^*) \leq n$.

Crucial: Flipping literal in unsatisfied clause c decreases d-value with probability at least $1/2$ (might even be bigger) since

- x^* satisfies c, hence
- at least one literal in c is differently assigned in x and x^*.

Using the fair random walk, d-value 0 is reached after expected $\leq n^2$ iterations. $\square$

Turning It into an RP-algorithm

Using $T = \infty$ means that algorithm does not stop on unsatisfiable instances. Instead use:

Theorem: RandomizedSat invoked with $T = 2n^2$ and $S = 1$ is an $\mathcal{RP}$-algorithm for $2\text{-}\textsc{Sat}$.

(Recall: $\mathcal{RP}$ has no false positives, success probability at least $1/2$.)

Proof: Nothing to show for unsatisfiable instances. If satisfiable, expected number $\boldsymbol{E}\left[\, I \,\right]$ of iterations fulfills $\boldsymbol{E}\left[\, I \,\right] \le n^2$. Use Markov's inequality (A.10 in lecture notes) to obtain $\boldsymbol{P}\left[\, I \ge 2\boldsymbol{E}\left[\, I \,\right] \right] \le 1/2$.

Analysis for 3-SAT (1/7)

RandomizedSat can be used on arbitrary SAT instances. Then we get the announced exponential running times:

Theorem: If given a satisfiable 3-SAT instance, RandomizedSat invoked with $T = 3n$ and $S = \infty$ outputs a satisfying assignment in expected time at most $p(n) \cdot (4/3)^n$, where $p(n)$ is a polynomial.

Proof idea: When an assignment is flipped, distance to satisfying assignment is decreased with probability at least $1/3$ ("unfair" random walk).

Reminder: Binomial Distribution

Suppose you do n trials, each of which independently is a success with probability p (e. g., roll a die n times and call a 6 a success).

Let X be the random number of successes.

Then X is called *binomially distributed* with parameters n and p. For $0 \leq k \leq n$, it holds

$$\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k},$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

is called binomial coefficient (how many ways are there of choosing k different objects from n different objects).

Analysis for 3-Sat (2/7)

Let d^* be the distance of initial guess and satisfying assignment. Note that $\boldsymbol{E}[d^*] = n/2$. To reach the goal, $2d^*$ decreasing and d^* increasing steps within $3d^*$ steps suffice.

This has probability at least

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

since the number of successes (decreasing steps) is binomially distributed with parameters $3d^*$ and $1/3$.

Analysis for 3-SAT (3/7)

There are many ways of bounding a binomial coefficient. We use:

$$\binom{\beta}{\alpha\beta} \geq \frac{1}{\beta+1} \cdot \left(\alpha^{\alpha} \cdot (1-\alpha)^{1-\alpha}\right)^{-\beta}.$$

because then (using $\alpha = 2/3$ and $\beta = 3d^*$)

$$\begin{aligned}
&\binom{3d^*}{2d^*}\left(\frac{1}{3}\right)^{2d^*}\left(\frac{2}{3}\right)^{d^*} \\
&\quad \geq \frac{1}{3d^*+1} \cdot \left(\left(\frac{1}{3}\right)^{1/3} \cdot \left(\frac{2}{3}\right)^{2/3}\right)^{-3d^*} \cdot \left(\frac{1}{3}\right)^{2d^*}\left(\frac{2}{3}\right)^{d^*} \\
&\quad \geq \cdots \geq \frac{1}{3n+1} \cdot \left(\frac{1}{2}\right)^{d^*}.
\end{aligned}$$

Analysis for 3-SAT (4/7)

Last expression is lower bound on the success probability but still depends on the random d^*. If $d^* = n$, it is very bad ($2^{-n}/(3n+1)$).

We take the average over the 2^n outcomes of the initial guess x_{in} and get a success probability of at least

$$\sum_y \boldsymbol{P}\left[x_{\text{in}} = y\right] \cdot \frac{1}{3n+1} \cdot \left(\frac{1}{2}\right)^{d(x^*,y)} = \frac{1}{3n+1} \boldsymbol{E}\left[\left(\frac{1}{2}\right)^{d(x^*,x_{\text{in}})}\right].$$

and are left with the expectation.

Since it counts the number of wrong bits in x_{in}, $d(x^*, x_{\text{in}}) = X_1 + \cdots + X_n$, where X_i independently uniformly distributed on $\{0, 1\}$.

Analysis for 3-Sat (5/7)

Hence,

$$\boldsymbol{E}\left[\left(\frac{1}{2}\right)^{d(x^*,x_{\mathsf{in}})}\right] = \boldsymbol{E}\left(\left(\frac{1}{2}\right)^{X_1+\cdots+X_n}\right)$$
$$= \boldsymbol{E}\left(\prod_{i=1}^{n}\left(\frac{1}{2}\right)^{X_i}\right) = \prod_{i=1}^{n}\boldsymbol{E}\left(\left(\frac{1}{2}\right)^{X_i}\right),$$

Note that $\boldsymbol{E}((1/2)^{X_i}) = (1/2)\cdot(1/2)^0 + (1/2)\cdot(1/2)^1 = 3/4$. Altogether, success probability is at least

$$\frac{1}{3n+1}\prod_{i=1}^{n}\frac{3}{4} = \frac{1}{3n+1}\left(\frac{3}{4}\right)^n.$$

Analysis for 3-SAT (6/7)

Each iteration of the outer loop is a success with probability at least

$$\frac{1}{3n+1}\left(\frac{3}{4}\right)^n.$$

Waiting time result from probability theory:
If an event occurs with probability p independently in every step, the expected number of steps until it occurs is $1/p$.

Therefore, expected number of iterations of outer loop is $O(n(4/3)^n)$.

Since $T = O(n)$, expected number of iterations is altogether $O(n^2(4/3)^n)$ and expected running time $O(p(n)(4/3)^n)$. $\square$

Analysis for 3-SAT (7/7)

Invoking RandomizedSat with $T = 3n$ and $S = 2p(n)(4/3)^n$ gives us a randomized algorithm with one-sided error and success probability at least $1/2$ in the positive case. (It's not an $\mathcal{RP}$-algorithm, why?)

Learned: Surprisingly simple randomized algorithms can be surprisingly good. Analysis requires several tools from probability theory.