## Best-SAT

YouTube

This post is an expanded translation of my lecture notes from a Randomized and Approximation Algorithms course that I took, and a more detailed explanation of the topics covered in my video about BEST-SAT.

## Basic definitions

Definition (Optimalization problem) is a tuple $\mathcal{I}, \mathcal{F}, f, g$

- set of all input instances $\mathcal{I}$
- sets of permissible inputs $\forall I \in \mathcal{I}: \mathcal{F}(I)$
- utility function $\forall I \in \mathcal{I}, A \in \mathcal{F}(I): f(I, A)$
- whether we're maximizing or minimizing (a single bit $g$ )

Definition (NP-Optimalization problem) is an optimalization problem $\mathcal{I}, \mathcal{F}, f, g$, for which we additionally require that:

- the length of all permissible solutions is polynomial
- the language of $(I, A), I \in \mathcal{I}, A \in \mathcal{F}(I)$ is polynomial
- we can check the correctness of a solution in polynomial time
- $f$ is computable in polynomial time

Definition: algorithm $A$ is $R$-approximation, if:

- it computes the solution in polynomial time (in terms of $|I|$ )
- for minimalization problem: $\forall I: f(A) \leq R \cdot \mathrm{OPT}(I)$
- for maximalization problem: $\forall I: f(A) \geq \mathrm{OPT}(I) / R$


## MAX-SAT

- Input: $C_{1} \wedge \ldots \wedge C_{n}$, each clause is a disjunction of $k_{j} \geq 1$ literals
- Output: evaluation $a \in\{0,1\}^{n}$ of the variables (sometimes called literals)
- Goal: maximize the number of satisfied clauses $\sum w_{j}$

We also assume that:

- no literal repeats in a clause
- at most one of $x_{i}, \bar{x}_{i}$ appearas in a clause


## RAND-SAT

## Algorithm (RAND-SAT)

1. choose all literals randomly (independently, for $p=1 / 2$ )
2. profit?

Theorem: RAND-SAT is a 2-approximation algorithm.
Proof: we'll create an indicator variable $Y_{j}$ for each clause

- the chance that $C_{j}$ is not satisfied is $\frac{1}{2^{k}}$

Since the size of the clause $k \geq 1$, we get $\mathbb{E}\left[Y_{j}\right]=\operatorname{Pr}\left[C_{j}\right.$ is satistied $]=1-\frac{1}{2^{k}} \geq \frac{1}{2}$, thus

$$
\mathbb{E}\left[\sum_{j=1}^{n} Y_{j}\right] \stackrel{\substack{\text { linearity } \\ \text { of expectation }}}{=} \sum_{j=1}^{n} \mathbb{E}\left[Y_{j}\right] \geq \sum_{j=1}^{n} \frac{1}{2} \geq \frac{1}{2} \mathrm{OPT}
$$

[^0]
## LP-SAT

## Algorithm (LP-SAT)

1. build an integer linear program:

- variables will be:
- $y_{i}$ for each literal
$-z_{j}$ for each clause
- inequalitites will be one for each clause, in the form

$$
z_{j} \leq \sum_{\text {positive }} y_{i}+\sum_{\text {negative }}\left(1-y_{i}\right)
$$

- we'll maximize the number of satisfied clauses $\sum z_{j}$

2. relax the program (allow real variables instead of integers) and calculate the optimum $y^{*}, z^{*}$
3. set literals $x_{i}$ to 1 with probability $y_{i}^{*}$

Theorem: LP-SAT is a $\left(1-\frac{1}{e}\right)$-approximation algorithm.
To prove this, we'll use a few lemmas/theorems that aren't difficult to prove, but aren't really interesting. I left links to (Wikipedia and I don't feel bad about it) articles with proofs for each, if you're interested.

## Fact (A - A/G mean inequality)

$$
\prod_{i=1}^{n} a_{i}^{\frac{1}{n}} \leq \frac{1}{n} \sum_{i=1}^{n} a_{i}
$$

Proof: https://en.wikipedia.org/wiki/Inequality_of_arithmetic_and_geometric_means
Fact (B - Jensen's inequality) if a function is concave on the interval $[0,1]$ and $f(0)=a, f(1)=a+b$, then

$$
\forall x \in[0,1]: f(x) \geq a+b x
$$

Proof: https://en.wikipedia.org/wiki/Jensen\'s_inequality
Fact (C - 1/e inequality)

$$
\left(1-\frac{1}{n}\right)^{n} \leq \frac{1}{e}
$$

Proof: https://en.wikipedia.org/wiki/E_(mathematical_constant)\#Inequalities

Proof (of the main theorem) consider $y^{*}, z^{*}$ and $C_{j}$ with $k_{j}$ literals; then

$$
\begin{aligned}
\operatorname{Pr}\left[C_{j} \text { is not satisfied }\right] & =\overbrace{\prod_{i: x_{i} \in C_{j}}\left(1-y_{i}^{*}\right)}^{\text {positive }} \overbrace{\prod_{i: \bar{x}_{i} \in C_{j}}}^{\text {negative }} y_{i}^{*} \\
& \leq\left[\frac{1}{k_{j}}\left(\sum_{i: x_{i} \in C_{j}}\left(1-y_{i}^{*}\right)+\sum_{i: \overline{x_{i} \in C_{j}}} y_{i}^{*}\right)\right]^{k_{j}} \\
& =\left[1-\frac{1}{k_{j}}\left(\sum_{i: x_{i} \in C_{j}} y_{i}^{*}+\sum_{i: \overline{x_{i} \in C_{j}}}\left(1-y_{i}^{*}\right)\right)\right]^{k_{j}} \\
& \leq\left(1-\frac{z_{j}^{*}}{k_{j}}\right)^{k_{j}}
\end{aligned}
$$

[^1]We're interested in the satisfied ones, so

$$
\begin{aligned}
\operatorname{Pr}\left[C_{j} \text { is satisfied }\right] & \geq 1-\left(1-\frac{z_{j}^{*}}{k_{j}}\right)^{k_{j}} \\
& \overbrace{\left.1-\left(1-\frac{1}{k_{j}}\right)^{k_{j}}\right] z_{j}^{*} \quad \begin{array}{l}
C \\
\geq \\
\text { our function } f\left(z_{j}^{*}\right) \\
\end{array}}=\left[1-\frac{1}{e}\right) z_{j}^{*}
\end{aligned}
$$

To use fact $B$, we observed that $a=f(0)=0$ and that the second derivation is non-positive (so the function is concave). Now to formally count how many our program satisfies:

$$
\begin{aligned}
\mathbb{E}\left[\sum_{j=1}^{m} Y_{j}\right] & =\sum_{j=1}^{m} \mathbb{E}\left[Y_{j}\right] \\
& \geq \sum_{j \in U} \operatorname{Pr}\left[C_{j} \text { is satisfied }\right] \\
& \geq \sum_{j \in U}\left(1-\frac{1}{e}\right) z_{j}^{*} \\
& =\left(1-\frac{1}{e}\right) \mathrm{OPT}
\end{aligned}
$$

## BEST-SAT

## Algorithm (BEST-SAT)

1. assign a value of a literal using RAND-SAT with probability $1 / 2$, else use BEST-SAT
2. have an existential crisis about the fact that this works and is asymptotically optimal

Theorem: BEST-SAT is $\frac{3}{4}$-approximation.
Proof: we want to prove that $\operatorname{Pr}\left[C_{j}\right.$ is satisfied $] \geq \frac{3}{4} z_{j}^{*}$.
Let's look at the probability that each algorithm satisfies a clause of $k$ variables:

- RAND-SAT: $1-\frac{1}{2^{k}}$ (at least one literal must be satisfied)
- LP-SAT: $\left[1-\left(1-\frac{1}{k}\right)^{k}\right] z_{j}^{*}$ (the formula right before using fact C)

Now the proof boils down to the following table:

| $k_{j}$ | RAND-SAT | LP-SAT | BEST-SAT |
| :--- | :--- | :--- | :--- |
| 1 | $\frac{1}{2} \geq \frac{1}{2} z_{j}^{*}$ | $1 \cdot z_{j}^{*}$ | $\frac{1}{2} \frac{1}{2}+\frac{1}{2} z_{j}^{*} \geq \frac{3}{4} z_{j}^{*}$ |
| 2 | $\geq \frac{3}{4} z_{j}^{*}$ | $\frac{3}{4} \cdot z_{j}^{*}$ | $\geq \frac{3}{4} z_{j}^{*}$ |
| $\geq 3$ | $\geq \frac{7}{8} z_{j}^{*}$ | $\geq\left(1-\frac{1}{e}\right) \cdot z_{j}^{*}$ | $>\frac{3}{4} z_{j}^{*}$ |


[^0]:    $[1]_{\text {An example problem could be minimum spanning trees: }}$

    - input instances: set of all weighted graphs
    - permissible inputs: spanning trees for the given weighted graph
    - utility function: the spanning tree weight (sum of its edges)
    - we're minimizing
    $[2]$ For minimalization problem, we ensure that the solution is always small enough. For maximalization problem, we ensure that the solution is always large enough.

[^1]:    $[3]$ We're using the optimal solution to the linear program (and generally the formula, if we allow real vlaues for literals) as a guide for our randomized algorithm.

