The Satisfiability Coding Lemma

Presenter: Harry Sha

March 2, 2022

Overview

CNFs

A CNF (Conjunctive Normal Form) formula is an “AND of ORs”. For example

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

Each of the “ORs” is called a clause. We'll use n for the number of variables and m for the number of clauses.

k-CNFs

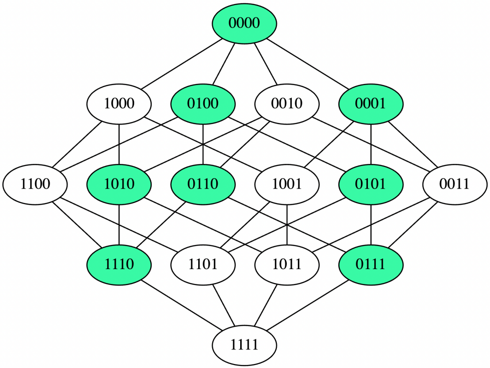
A k-CNF is one where each clause has at most k literals.

We care a lot about k-CNFs because the problem of whether or not a k-CNF is satisfiable is NP-Complete (for $k \geq 3$).

We'll call the satisfying assignments solutions.

Structure of solutions

What can we say about the structure of solutions?



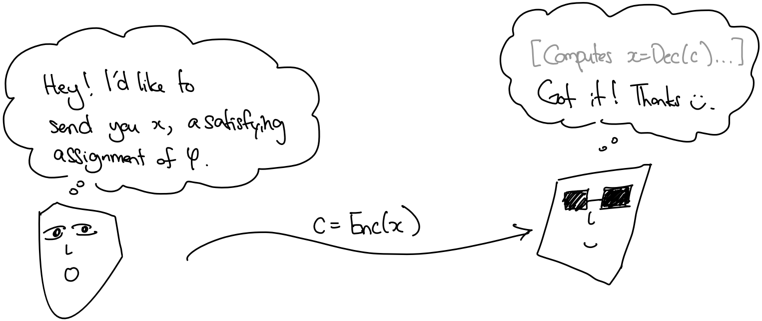
$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

Today

In this talk I'll present the *Satisfiability Coding Lemma* of Ramamohan Paturi, Pavel Pudlak, and Francis Zane which sheds light on the structure of solutions of k-CNFs.

These insights will give rise to "fast" (still exponential time!) algorithms for k-SAT, and are crucial in state-of-the-art SAT algorithms.

SAT and Coding



hopefully $\mathrm{Enc}(x)$ can be shorter than x so we can save on communication

Encoding

Idea: Send 1 bit at a time. “Substitute” the value of a variable after every bit. If at any point, a variable is “forced”, you don't need to send it.

Encoding

Idea: Send 1 bit at a time. "Substitute" the value of a variable after every bit. If at any point, a variable is "forced", you don't need to send it.

Substitute: $x_i \to 1$ means replace all the x_i's with 1 and all the $\neg x_i$'s with 0s (The opposite for $x_i \to 0$). Then, get rid of all the clauses that have a 1, and all literals which are assigned 0.

Forced: A variable x_i is forced if either x_i or $\neg x_i$ is a clause (of size one).

Example

$$\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

$x = 1110$

① Send $x_1 = 1$

$$\varphi = \cancel{(x_1 \vee x_2 \vee \neg x_3)} \wedge (\cancel{\neg x_1} \vee x_2 \vee \neg x_4) \wedge (\cancel{\neg x_1} \vee \neg x_2 \vee \neg x_4) \wedge (\cancel{\neg x_1} \vee x_3 \vee x_4)$$

$$\downarrow$$

$$\varphi = (x_2 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_3 \vee x_4).$$

$$\downarrow$$

② Send $x_2 = 1$, $\quad \varphi = (\neg x_4) \wedge (x_3 \vee x_4)$.

$$\downarrow$$

③ $x_4 = 0$ is forced! $\quad \varphi = (x_3)$

④ $x_3 = 1$ is forced!

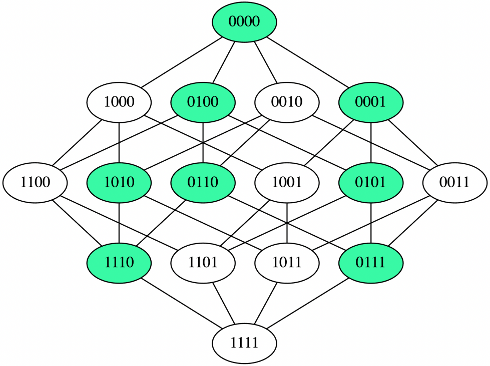
We only sent two bits :)

When does this work well?

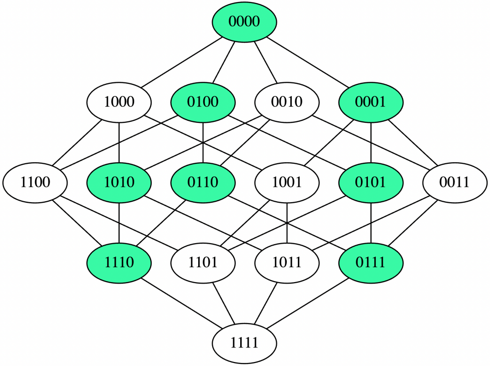
Let $x \in \{0,1\}^n$ be a satsifying assignment. We don't need to send $x[i]$, the assignment of variable x_i, if at some point, x_i or $\neg x_i$ became a unit clause. This happens when both of the following are true

1. There is a clause C where the literal involving x_i is the only literal assigned 1. We say that C is critical for x_i with respect to x.
2. x_i is the 'last' variable in C i.e. for $x_j \in C$, $j \leq i$.

Critical clauses



Critical clauses



x_i has a critical clause $\iff$ flipping the ith bit of x results in an unsatisfying assignment.

j-Isolated solutions

For this reason, if a solution $x \in \{0,1\}^n$ has j variables with critical clauses, we call it j-isolated. I.e. j of x's n neighbors are unsatisfying.

Getting 'lucky'

Remember that in order to skip a variable's assignment, we need it to occur 'last' in the its critical clause. To address this, first shuffle the order of the variables.

Define Enc_π to be the encoding that first permutes its variables according to the permutation π and then applies the function from slide 8.

The Satsifability Coding Lemma

Here is the Satisfiability Coding Lemma (SCL)

Theorem (Satisfiability Coding Lemma)

Let φ be any k-CNF over n variables. If x is a j-isolated solution of φ, then the average (over permutations π) encoding length of x under the Enc_π is at most $n - j/k$.

Proof.

Proof: Let I be the set of variables w/ critical clauses, note $|I| = j$.

$$\mathop{\mathbb{E}}_{\sigma \sim S_n}[\#\text{ of forced variables}] = \mathop{\mathbb{E}}_{\sigma \sim S}\left[\sum_{i \in I} \mathbb{1}\{x_i \text{ is forced}\}\right]$$

$$\geq \mathop{\mathbb{E}}_{\sigma \sim S_n}\left[\sum_{i \in I} \mathbb{1}\{x_i \text{ is last in its clause wrt } \sigma\}\right] \quad \text{(this is what it means to be forced)}$$

$$= \sum_{i \in I} \Pr(x_i \text{ is last in its clause}) \quad \text{(linearity of expectation).}$$

$$\geq \sum_{i \in I} 1/k \quad \text{(every clause has} \leq k \text{ literals)}$$

$$= j/k.$$

So we get to delete at least j/k bits on average.

$\Rightarrow$ average description length $\leq n - j/k$.

∎

The number of n-isolated solutions

A n-isolated solution has no neighboring solutions. We can use the SCL to show:

Theorem

A k-CNF over n variables has at most $2^{n-n/k}$ isolated solutions.

Fact: if $\Phi : S \to \{0,1\}^*$ is a deterministic prefix free encoding with $\underbrace{\text{average encoding length } \ell}_{\text{average over inputs } s \in S}$, $|S| \le 2^\ell$.

this is Fact 1 in the paper, you can check it for the proof! I believe it's pretty standard...

Proof: We'll show for <u>some</u> permutation σ, the average encoding of any isolated solution is $\le n - n/k$.

Let's compute the average (over permutations) average encoding length (over isolated solns). Denote the set of isolated solns by I.

$$\mathop{E}_{\sigma \sim S_n}\left[\mathop{E}_{x \sim I}\left[|\Phi_\sigma(x)| \right]\right] = \frac{1}{n!}\sum_{\sigma \in S_n} \frac{1}{|I|}\sum_{x \in I} |\Phi_\sigma(x)|$$

$$= \frac{1}{|I|}\sum_{x \in I} \frac{1}{n!}\sum_{\sigma \in S_n} |\Phi_\sigma(x)| \qquad \text{(Exchange sums)}$$

$$\le \frac{1}{|I|}\sum_{x \in I} n - n/k \qquad \text{(SCL)}.$$

$$= n - n/k.$$

Thus, some permutation σ exists s.t. the average encoding length of isolated sols $\le n - n/k$ "first moment method". ∎

This bound is tight - example: block parity

Let $n = sk$ for some integer s. Break up the variable in to s groups of k variables. Let F_i be the k-CNF computing parity on the k variables of the ith group. Let

$$F = F_1 \wedge F_2 \wedge ... \wedge F_s.$$

There are $2^{s(k-1)} = 2^{n-n/k}$ solutions for F. Furthermore, every solution of F has the same parity and are thus n-isolated.

The algorithm

Algorithm 1: Randomized algorithm for SAT

Input: φ

while *Some variable is not yet assigned* **do**
  if *a variable* x_i *is fixed* **then**
    assign x_i the forced value and substitute in φ
  else
    $x_i =$ random unassigned variable
    $b =$ random pick from $\{0, 1\}$
    assign x_i to b and substitute b for x_i in φ.
  end
end
If the assignment we found is satisfying, return it!

Success probability

Theorem

If φ is a satisfiable k-CNF over n variables. The algorithm in the previous slide finds a satisfying assignment with probability at least

$$\frac{1}{n2^{n-n/k}}$$

Note that this algorithm runs in time $|\varphi|$, so repeating the algorithm $n^2 2^{n-n/k}$ times we find a satisfying assignment with probability approaching 1. The overall runtime of the algorithm is then $O(n^2|\varphi|2^{n-n/k})$.

Proof: Let x be a j-isolated satisfying assignment of φ. We first lowerbound the probability that the algorithm outputs x.

Define: E_1: at least j/k variables occur at the end of their critical clauses.

E_2: whenever we randomly assign the value, it agrees with our assignment.

Note $\Pr[\text{Alg outputs } x] \geq \Pr[E_1 \wedge E_2] = \underbrace{\Pr[E_1]}\,\underbrace{\Pr[E_2 \mid E_1]}$.

We now lower bound these probabilities.

Claim $\Pr[E_1] \geq 1/n$.

Let A be the # of variables that occur last in their critical clause. Note $E[A] = j/k$. Since A is at most j, $\Pr[A \geq j/k] \geq 1/k$ (worst case A is always 0 or j).

$$\geq 1/n$$

Claim: $\Pr[E_2 \mid E_1] \geq 2^{-n+j/k}$

Since E_1 holds, we get to skip j/k variables, we need to get lucky for the remaining $n-j/k$ variables.

Thus, in total, $\Pr[\text{Alg outputs } x] \geq 1/n \cdot 2^{-n-j/k}$.

Now, we sum over all satisfying assignments, x. Let S be the set of satisfying assignments and $I: S \to \{0,1,\dots n\}$ map x to the "isolation of x."

$$\Pr[\text{Alg works}] = \sum_{x \in S} \Pr[\text{Alg outputs } x]$$

$$= \sum_{x \in S} \tfrac{1}{n} 2^{-n + I(x)/k}$$

$$= \frac{1}{n} \cdot 2^{-n+n/k} \sum_{x \in S} 2^{(I(x)-n)/k}$$

$$\geq \frac{1}{n} \cdot 2^{-n+n/k} \sum_{x \in S} 2^{I(x)-n}$$

we're good if this is ≥ 1!

Claim: $\sum_{x \in S} 2^{I(x)-n} \geq 1$ for all non-empty subsets $S \subseteq \{0,1\}^n$

Let $N_S(x)$ denote the set of neighbors of x in S. The expression is then $\sum_{x \in S} (1/2)^{|N_S(x)|}$

By induction on n: Base case. $n=0$ ✓. S_i subset of S that starts w/ a i

Let $S \subseteq \{0,1\}^n$. Fix $S_1 = 1\{0,1\}^* \cap S$, $S_0 = 0\{0,1\}^* \cap S$

Case 1 S_0, S_1 non empty. $\sum_{x \in S} (1/2)^{|N_S(x)|} = \sum_{x \in S_0} (1/2)^{|N_S(x)|} + \sum_{x \in S_1} (1/2)^{|N_S(x)|}$

$$\geq \sum_{x \in S_0} (1/2)^{|N_{S_0}(x)|+1} + \sum_{x \in S_1} (1/2)^{|N_{S_1}(x)|+1}$$

$$\geq 1/2 + 1/2 \quad \text{(IH).}$$

Case 2 S_0 or S_1 is empty. $\sum_{x \in S} (1/2)^{|N_S(x)|} = \sum_{x \in S_0} (1/2)^{|N_S(x)|} = \sum_{x \in S_0} (1/2)^{|N_{S_0}(x)|} \underset{\text{(IH)}}{\geq} 1$

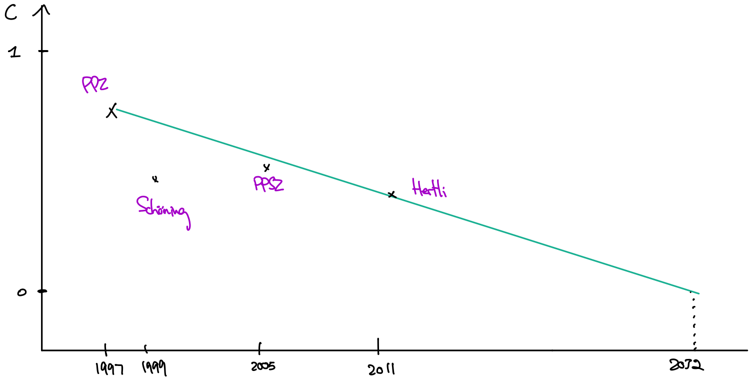
↑ x has no neighbors that start w/ 1!

Aside: (randomized) SAT algorithms since PPZ

- $2^{0.667n}$ Satisfiability Coding Lemma (PPZ, 1997).
- $2^{0.415n}$ A Probabilistic Algorithm For k-SAT and Constraint Satisfaction Problems (Schöning, 1999). (only better for k=3)
- $2^{0.521n}$ An Improved Exponential-Time Algorithm for k-SAT (PPSZ, 2005).
- $2^{0.387n}$ 3-SAT Faster and Simpler - Unique-SAT Bounds for PPSZ Hold in General (Hertli, 2011)
- Some tiny (but meaningful) improvements have been made since.
- State of the art: Faster k-SAT Algorithms using Biased-PPSZ (HKZZ, 2019)

A case for P=NP.

"$\exists$ a randomized alg for 3-SAT in time 2^{cn}"



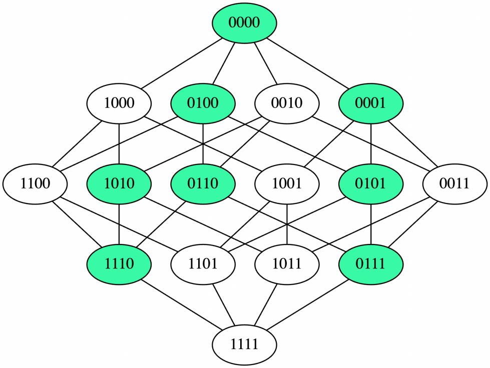
Derandomization

You can derandomize this algorithm using k-wise independence.

The idea is to try every permutation in a smaller space of permutations and prove a similar bound on probability using k-wise independence.

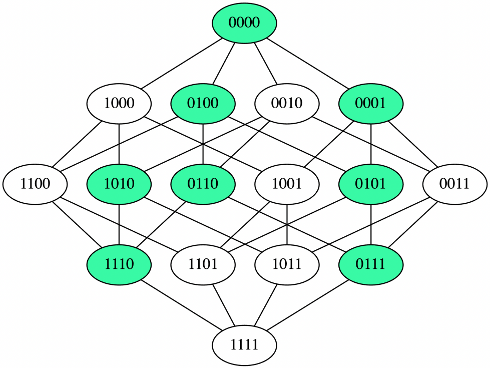
How would you find the smallest equivalent DNF?

A DNF is an OR of ANDs. We're looking for the minimal number of ANDs in a DNF.



$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

Implicants



$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee x_4)$$

An implicant is a set of literals that imply the formula. They correspond to a subcube of satisfying assignments. For example, $\{\neg x_1, \neg x_3\}$ is an implicant here, corresponding to the square of solutions $\{0000, 0100, 0001, 0101\}$

Prime implicants and subcubes

Bigger cubes consist of smaller cubes, instead of taking many smaller cubes, we'd rather take the bigger cube.

A prime implicant is an subcube of solutions that is not contained within a larger cube. Equivalently, when viewed as a set of literals, an implicant is prime if no strict subset is also an implicant.

Prime implicants are interesting and well studied things.

Another view of implicants

An equivalent way to define an implicant is a partial assignment which we can view as a n-bit string $I = \{0, 1, *\}^n$. If I is an implicant, no matter how you fill in the $*$s, you get a satisfying assignment.

Equivalently, every clause has a literal assigned 1.

Critical clauses and isolation for implicants

For an implicant I in $\{0, 1, *\}^n$ of φ, a clause C is critical for x_i if the literal involving x_i is the only literal in C assigned 1 (the rest are either 0 or *).

In this case if you flipping the assignment of x_i makes I no longer an implicant - there is some way of assigning the free variables to falsify the clause C!

As before, an implicant is j-isolated if j variables have critical clauses.

Definition check: Prime implicants of size j are j-isolated

Definition check: Prime implicants of size j are j-isolated

Let I be a prime implicant of size j. If I is not j-isolated, there is some variable x_i that is fixed by I (not assigned *) that does not have a critical clause. Then we claim $I \setminus \{x_i\}$ is a smaller implicant. Indeed since x_i did not have a critical clause, any clause for which the literal involving x_i was assigned 1, has another literal assigned 1! So each clause still has at least one literal assigned 1 and $I \setminus \{x_i\}$ is still an implicant.

Encoding Implicants

There is a natural generalization of the encoding algorithm for satisfying assignments to implicants. For any clause, if all the literals in the clause so far have been assigned * or 0, the last one must be a 1. So again the algorithm looks like the following.

Encoding Implicants

There is a natural generalization of the encoding algorithm for satisfying assignments to implicants. For any clause, if all the literals in the clause so far have been assigned * or 0, the last one must be a 1. So again the algorithm looks like the following.

- Send 1 bit in $\{0, 1, *\}$ at a time.
- Substitute the values of the variable in to φ.
- If at any point a variable is forced, you don't need to send it!

The only difference here is we are sending bits in $\{0, 1, *\}$. To substitute $*$ for a variable x_i in the formula, we simply mean to remove all literals x_i and $\neg x_i$.

Implicant coding lemma

Theorem (Implicant coding lemma)

Let φ be any k-CNF over n variables. If x is a j-isolated implicant of φ, then the average (over permutations π) encoding length of x under the Enc_π is at most $n - j/k$.

Applications

- The number of size-j prime implicants of a k-CNF is at most $3^{n-j/k}$.
- The number of prime implicants of a k-CNF is at most $3^{n(1-\Omega(1/k))}$. Previous best, $3^n/\sqrt{n}$ for general CNFs, $3^{n(1-\Omega(1/rk))}$ for read r. Best lower bound is $3^{n(1-\frac{O(\log(k))}{k})}$
- We can enumerate the set of prime implicants of a k-CNF in time $O(|\varphi| \cdot n^{2k+2} \cdot 3^{n(1-c/k)})$.
- The number of size-j prime implicants of a m-clause CNFs is at most $3^{n(1-\Omega(j/\log(m)))}$.
- The number of prime implicants of a monotone k-CNF is at most $2^{n(1-\Omega(1/k))}$.
- The smallest DNF size for a monotone k-CNF is at most $n2^{n(1-0.282/k)}$.