A #SAT Algorithm for Small-size Constant Depth PTF circuits

Swapnam Bajpai, Vaibhav Krishan, Nutan Limaye, Deepanshu Kush, & Srikanth Srinivasan

IIT Bombay

Introduction

$f : \{-1,1\}^n \to \{-1,1\}$ is a ***k*-PTF** if there exists a multilinear polynomial P of degree k such that for all $a \in \{-1,1\}^n$, $f(a) = \mathrm{sgn}(P(a))$.

Input parameters:

- n, the number of variables,
- $M = \log_2(\Sigma|\alpha_i|)$, where $\alpha_i \in \mathbb{Z}$ are coefficients of P.

***k*-PTF-SAT**	Given a degree k polynomial P with (n, M) as parameters, does there exist $a \in \{-1,1\}^n$, $P(a) < 0$. Counting $a \in \{-1,1\}^n$ such that $P(a) < 0$ is **#*k*-PTF-SAT**.

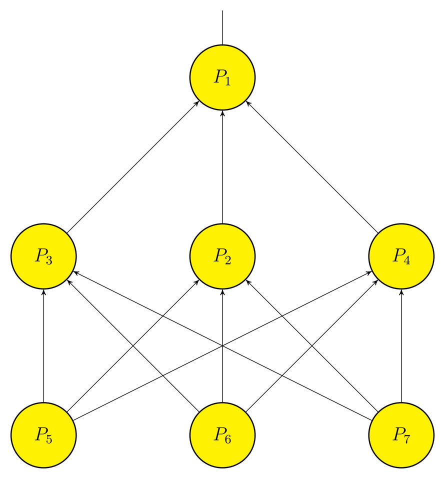

Fig. 1: k-PTF circuits, $\deg(P_i) \le k$

Input parameters:

- n, the number of variables,
- $M = \max_i \log_2(\Sigma_j|\alpha_{ij}|)$, α_{ij} coefficients of P_i,
- s is the number of wires,
- d is the number of layers (depth).

SAT for *k*-PTF circuits	Given a k-PTF circuit with parameters (n, M, s, d), does there exist $a \in \{-1,1\}^n$ such that $C(a) = -1$. Counting $a \in \{-1,1\}^n$ such that $C(a) = -1$ is #SAT for k-PTF circuits.

Motivations

- For many circuit classes $\mathcal{C}$, analogously defined $\mathcal{C} - SAT$ is studied widely.
- Trivial algorithm run in time $2^n \mathrm{poly}(|C|)$,
- Even superpolynomial savings imply superpolynomial **lower bounds** against NEXP.
- Connections to combinatorial problems.
- PTFs naturally generalize study of perceptrons.

Prior Work

- 2-PTF-SAT in time $2^{n-\Omega(\sqrt{n})}$ by Williams (ICALP'04, STOC'14).
- #k-PTF-SAT for $M \le O(n^{1-\Omega(1)})$ by Sakai et al.
- #SAT for PTF circuits of small size and with few monomials in each polynomial, running in time $\mathrm{poly}(n, M)2^{n-n^{\varepsilon}}$ by Kabanets and Lu (inspired by lower bound for small PTF circuits by Kane, Kabanets, and Lu).

Open until now	• #2-PTF-SAT. • k-PTF-SAT for $k \ge 3$. • #SAT for k-PTF circuits.

Online link

https://arxiv.org/abs/1809.05932

Results and Techniques

Result 1	For $k = O(1)$, there is a zero-error randomized algorithm for #k-PTF-SAT with parameters (n, M) which runs in time $\mathrm{poly}(n, M) \cdot 2^{n-\tilde{\Omega}(S)}$, where $S = n^{1/(k+1)}$.

Proof Sketch:

- Follows the outline of Sakai et al.
  - *Step 1*: Count and store the number of satisfying assignments for each PTF on *few* variables.
  - *Step 2*: For all but a *few* variables, consider all the reduced PTFs by assigning these chosen variables all possible values. For each such PTF use the already stored value and sum them all up.
- *Crucial change*: Use a learning algorithm by Kane, Lovett, Moran, and Zhang in step 1.

Result 2	For constant k, d, there exists a constant $\varepsilon_{k,d} > 0$ such that, there is a zero-error randomized algorithm that solves the #SAT problem for k-PTF circuits of size at most $s = n^{1+\varepsilon_{k,d}}$ with high probability. The algorithm runs in time $\mathrm{poly}(n, M) \cdot 2^{n-S}$, where $S = n^{\varepsilon_{k,d}}$ and (n, s, d, M) are the parameters of the input k-PTF circuit.

Proof Sketch:

- Follows the outline of Kabanets and Lu.
- Inductive proof.
  - *Base Case*: Solve the problem for a small AND of k-PTFs.
  - *Inductive Case*:
    1. Apply random restrictions,
    2. Many gates at the bottom layer become biased towards a specific output,
    3. Check all the assignments which make these gates take the *minority* value,
    4. Replace these gates with their *majority* values and remember this by moving them under an AND,
    5. For other gates, replace with all possible values and remember this by moving them under an AND,
    6. Recurse on lower depth circuits thus formed.
- *Crucial changes*:
  - Use our algorithm from the previous result for base case.
  - Enumerating the assignments which lead to *minority* output also uses previous result.

Further work

- Derandomizing the algorithm (promising leads exist)?
  - Would give lower bounds against NEXP.
  - But Kane, Kabanets, and Lu already proved it.
- Improving the constraints on the number of wires?
  - Note that there are bootstrapping results.
  - But we don't achieve the requirements.
- Extending to growing degrees?
  - Note that the algorithm from Kabanets and Lu works for any degree function.
  - But their sparsity constraint is equivalent to degree being constant.