A #SAT Algorithm for Small Constant-Depth Circuits with PTF gates

Swapnam Bajpai[1] Vaibhav Krishan[1] Deepanshu Kush[2]
Nutan Limaye[1] Srikanth Srinivasan[2]

[1]Department of Computer Science and Engineering, IIT Bombay, Mumbai, India

[2]Department of Mathematics, IIT Bombay, Mumbai, India

Innovations in Theoretical Computer Science 2019

Circuits

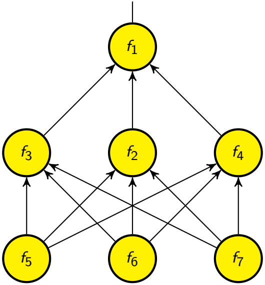

Figure: Circuits

-1 is *true*, 1 is *false*.

Circuit satisfiability

- $\mathcal{C}$-**SAT**: Given a circuit C on n inputs from a circuit class $\mathcal{C}$, check if there exists $a \in \{-1, 1\}^n$ such that $C(a) = -1$.
- $\#\mathcal{C}$-**SAT**: Count $a \in \{-1, 1\}^n$ such that $C(a) = -1$.
- Trivial algorithm in time $2^n \operatorname{poly}(|C|)$.
- Williams proved (co-non)deterministic $\mathcal{C}$-SAT algorithms running in time $O(2^n/n^{\omega(1)})$ imply $\mathrm{NEXP} \not\subset \mathcal{C}$.
  - This was used to prove $\mathrm{NEXP} \not\subset \mathrm{ACC}$.

Circuit satisfiability

- $\mathcal{C}$-**SAT**: Given a circuit C on n inputs from a circuit class $\mathcal{C}$, check if there exists $a \in \{-1, 1\}^n$ such that $C(a) = -1$.
- #$\mathcal{C}$-**SAT**: Count $a \in \{-1, 1\}^n$ such that $C(a) = -1$.
- Trivial algorithm in time $2^n \operatorname{poly}(|C|)$.
- Williams proved (co-non)deterministic $\mathcal{C}$-SAT algorithms running in time $O(2^n/n^{\omega(1)})$ imply $\mathrm{NEXP} \not\subset \mathcal{C}$.
  - This was used to prove $\mathrm{NEXP} \not\subset \mathrm{ACC}$.

Circuit satisfiability

- $\mathcal{C}$-**SAT**: Given a circuit C on n inputs from a circuit class $\mathcal{C}$, check if there exists $a \in \{-1, 1\}^n$ such that $C(a) = -1$.
- #$\mathcal{C}$-**SAT**: Count $a \in \{-1, 1\}^n$ such that $C(a) = -1$.
- Trivial algorithm in time $2^n \operatorname{poly}(|C|)$.
- Williams proved (co-non)deterministic $\mathcal{C}$-SAT algorithms running in time $O(2^n/n^{\omega(1)})$ imply $\mathrm{NEXP} \not\subset \mathcal{C}$.
  - This was used to prove $\mathrm{NEXP} \not\subset \mathrm{ACC}$.

Circuit satisfiability

- $\mathcal{C}$-**SAT**: Given a circuit C on n inputs from a circuit class $\mathcal{C}$, check if there exists $a \in \{-1,1\}^n$ such that $C(a) = -1$.
- #$\mathcal{C}$-**SAT**: Count $a \in \{-1,1\}^n$ such that $C(a) = -1$.
- Trivial algorithm in time $2^n \operatorname{poly}(|C|)$.
- Williams proved (co-non)deterministic $\mathcal{C}$-SAT algorithms running in time $O(2^n/n^{\omega(1)})$ imply $\mathrm{NEXP} \not\subset \mathcal{C}$.
    - This was used to prove $\mathrm{NEXP} \not\subset \mathrm{ACC}$.

Definitions and Notations

Definition (k-PTF)

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

Definitions and Notations

Definition (k-PTF)

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

Important parameters are:

- n is the number of inputs,
- $M = \lceil \log_2(\sum |\alpha_i|) \rceil$, $\alpha_i \in \mathbb{Z}$ are coefficients in P.

Definition (k-PTF-SAT)

Given a polynomial P with parameters (n, M), does there exist $a \in \{-1, 1\}^n$ such that $P(a) < 0$.

Example

Let $P = 2x_1x_2 + 4x_2x_3 - 3x_1x_3$.

Example

Let $P = 2x_1x_2 + 4x_2x_3 - 3x_1x_3$. Then $f : \{-1,1\}^3 \to \{-1,1\}$ be defined by the truth table:

x_1	x_2	x_3	f
1	1	1	1
1	1	−1	1
1	−1	1	−1
1	−1	−1	1
−1	1	1	1
−1	1	−1	−1
−1	−1	1	1
−1	−1	−1	1

Example

Let $P = 2x_1x_2 + 4x_2x_3 - 3x_1x_3$. Then $f : \{-1, 1\}^3 \to \{-1, 1\}$ be defined by the truth table:

x_1	x_2	x_3	f
1	1	1	1
1	1	−1	1
1	−1	1	−1
1	−1	−1	1
−1	1	1	1
−1	1	−1	−1
−1	−1	1	1
−1	−1	−1	1

Then f is a 2-PTF, defined by P.

Example

Let $P = 2x_1x_2 + 4x_2x_3 - 3x_1x_3$. Then $f : \{-1,1\}^3 \to \{-1,1\}$ be defined by the truth table:

x_1	x_2	x_3	f
1	1	1	1
1	1	-1	1
1	-1	1	-1
1	-1	-1	1
-1	1	1	1
-1	1	-1	-1
-1	-1	1	1
-1	-1	-1	1

Then f is a 2-PTF, defined by P.

$n = 3$ and $M = \lceil \log_2(9) \rceil = 4$.

Prior work

- 2-PTF-SAT in time $2^{n-\Omega(\sqrt{n})}$ by Williams (ICALP'04, STOC'14).
- #k-PTF-SAT for $M \leq O(n^{1-\Omega(1)})$ by Sakai et al.
- Open until our work:
  - #2-PTF-SAT.
  - k-PTF-SAT for $k \geq 3$.

Prior work

- 2-PTF-SAT in time $2^{n-\Omega(\sqrt{n})}$ by Williams (ICALP'04, STOC'14).
- $\#k$-PTF-SAT for $M \leq O(n^{1-\Omega(1)})$ by Sakai et al.
- Open until our work:
  - $\#$2-PTF-SAT.
  - k-PTF-SAT for $k \geq 3$.

Prior work

- 2-PTF-SAT in time $2^{n-\Omega(\sqrt{n})}$ by Williams (ICALP'04, STOC'14).
- $\#k$-PTF-SAT for $M \leq O(n^{1-\Omega(1)})$ by Sakai et al.
- Open until our work:
  - $\#$2-PTF-SAT.
  - k-PTF-SAT for $k \geq 3$.

Our results

Theorem

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

Two steps:

1. Solve and store the answer for all k-PTFs on m variables.
2. For each partial assignment $\{x_{m+1}, \ldots, x_n\} \to \{-1, 1\}$, apply the partial assignment and use the stored answers.

Appropriate value for m gives the desired runtime. Approach is similar to Sakai et al.

Crucial difference is, we use learning algorithms designed by Kane, Lovett, Moran, Zhang in step 1.

Proof sketch

Two steps:

1. Solve and store the answer for all k-PTFs on m variables.
2. For each partial assignment $\{x_{m+1}, \ldots, x_n\} \to \{-1, 1\}$, apply the partial assignment and use the stored answers.

Appropriate value for m gives the desired runtime. Approach is similar to Sakai et al.

Crucial difference is, we use learning algorithms designed by Kane, Lovett, Moran, Zhang in step 1.

Proof sketch

Two steps:

1. Solve and store the answer for all k-PTFs on m variables.
2. For each partial assignment $\{x_{m+1}, \ldots, x_n\} \to \{-1, 1\}$, apply the partial assignment and use the stored answers.

Appropriate value for m gives the desired runtime. Approach is similar to Sakai et al.

Crucial difference is, we use learning algorithms designed by Kane, Lovett, Moran, Zhang in step 1.

Proof sketch

Two steps:

1. Solve and store the answer for all k-PTFs on m variables.
2. For each partial assignment $\{x_{m+1}, \ldots, x_n\} \to \{-1, 1\}$, apply the partial assignment and use the stored answers.

Appropriate value for m gives the desired runtime. Approach is similar to Sakai et al.

Crucial difference is, we use learning algorithms designed by Kane, Lovett, Moran, Zhang in step 1.

Proof sketch

Two steps:

1. Solve and store the answer for all k-PTFs on m variables.
2. For each partial assignment $\{x_{m+1}, \ldots, x_n\} \to \{-1, 1\}$, apply the partial assignment and use the stored answers.

Appropriate value for m gives the desired runtime. Approach is similar to Sakai et al.

Crucial difference is, we use learning algorithms designed by Kane, Lovett, Moran, Zhang in step 1.

More results

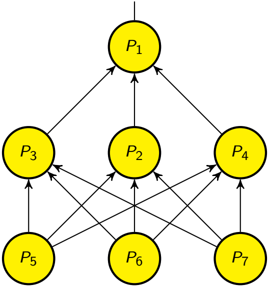

Figure: k-PTF circuits, $\max_i \deg(P_i) \leq k$

- #SAT for small k-PTF circuits but with sparsity restriction by Kabanets and Lu (inspired by Kane, Kabanets, and Lu).
- We give #SAT algorithm for small k-PTF circuits as well.

More results

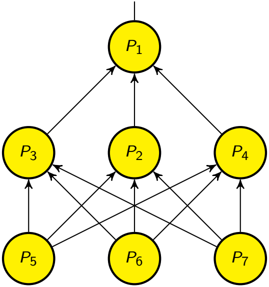

Figure: k-PTF circuits, $\max_i \deg(P_i) \leq k$

- #SAT for small k-PTF circuits but with sparsity restriction by Kabanets and Lu (inspired by Kane, Kabanets, and Lu).
- We give #SAT algorithm for small k-PTF circuits as well.

More results

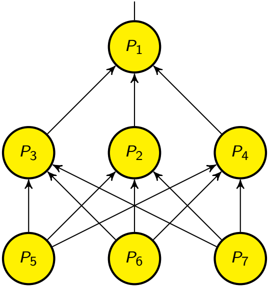

Figure: k-PTF circuits, $\max_i \deg(P_i) \leq k$

- #SAT for small k-PTF circuits but with sparsity restriction by Kabanets and Lu (inspired by Kane, Kabanets, and Lu).
- We give #SAT algorithm for small k-PTF circuits as well.

THANK YOU.......

DO YOU HAVE ANY QUESTİONS ?