The Polynomial Method in Circuit Complexity, Applied to Algorithm Design

Function $f$ doesn’t have “nice” polynomials

Randomized reduction

Low complexity

“nice” polynomials

“classical” algorithm

Multipoint evaluation of polynomials (MM/FFT) ➔ faster algorithm!

Ryan Williams

Stanford
Outline

• The Polynomial Method and Circuit Lower Bounds
• All-Pairs Shortest Paths in Dense Graphs
• Other Algorithmic Applications
• Future Work
Lower Bounds for Constant-Depth Circuits

Ajtai, Furst-Saxe-Sipser (early 80’s)
The MOD2 function on \( n \) bits is not computable in AC0

That is, PARITY can’t be computed by constant depth circuits of polynomial size over AND, OR, NOT

Cannot compute PARITY in constant parallel time with polynomially many “OR/AND/NOT” operations

KEY IDEA: “RANDOM RESTRICTION” [Subbottovskaya ’60s]
If you randomly select \( n - \sqrt{n} \) inputs and randomly set them to 0-1 values, chances are that the whole circuit will simplify to a constant (either 0 or 1).
Lower Bounds for Constant-Depth Circuits

Ajtai, Furst-Saxe-Sipser (early 80’s)

The MOD2 function on n bits is not computable in AC0

That is, PARITY can’t be computed by constant depth circuits of polynomial size over AND, OR, NOT

Cannot compute PARITY in constant parallel time with polynomially many “OR/AND/NOT” operations

KEY IDEA: “RANDOM RESTRICTION” [Subbottovskaya ’60s]

If you randomly select $n - \sqrt{n}$ inputs and randomly set them to 0-1 values, chances are that the whole circuit will simplify to a constant (either 0 or 1).
Lower Bounds for Constant-Depth Circuits

Ajtai, Furst-Saxe-Sipser (early 80’s)

The MOD2 function on n bits is not computable in AC0

That is, PARITY can’t be computed by constant depth circuits of polynomial size over AND, OR, NOT

Cannot compute PARITY in constant parallel time with polynomially many “OR/AND/NOT” operations

KEY IDEA: “RANDOM RESTRICTION” [Subbotovskaya ’60s]
If you randomly select $n - \sqrt{n}$ inputs and randomly set them to 0-1 values, chances are that the whole circuit will simplify to a constant (either 0 or 1).
Lower Bounds for Constant-Depth Circuits

Ajtai, Furst-Saxe-Sipser (early 80’s)
The MOD2 function on n bits is not computable in AC0

That is, PARITY can’t be computed by constant depth circuits of polynomial size over AND, OR, NOT

Cannot compute PARITY in constant parallel time with polynomially many “OR/AND/NOT” operations

KEY IDEA: “RANDOM RESTRICTION” [Subbotovskaya ’60s]
If you randomly select $n - \sqrt{n}$ inputs and randomly set them to 0-1 values, chances are that the whole circuit will simplify to a constant (either 0 or 1).

PARITY only simplifies to a constant when all variables are set!
Lower Bounds for Constant-Depth Circuits

Razborov, Smolensky (late 80’s)

MAJORITY $\notin \text{AC0}[\oplus] = \text{AC0}$ with MOD2 gates
MOD3 $\notin \text{AC0}[\oplus]$

Random Restriction no longer works!

MOD2 (PARITY) only simplifies to a constant after all inputs are assigned
PARITY Circuits Have “Small” Polynomials

Toy Theorem: No circuit of PARITY gates can compute the AND function

Every function computed by a circuit made up of PARITY gates can be represented by a multivariate polynomial over $\mathbb{F}_2$ of degree 1.

However, the AND function on $n$ variables has degree $n$ over $\mathbb{F}_2$!

$$\text{AND}(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot x_3 \cdot x_4$$
The Polynomial Method in Circuit Complexity

Want to show:
function F does not have “low complexity” circuits

1. Convert circuits of “low” complexity...

... into polynomials of “low” degree

2. Show F can’t be modeled by “low” degree polynomials
Reduce $\text{AC0}[\oplus]$ to simple polynomials

**Theorem [Razborov-Smolensky’87]** For every $\text{AC0}[\oplus]$ circuit $C$ with $n$ inputs, size $s$, and depth $d$, there is an efficiently samplable distribution $D(C)$ of polynomials of degree $(\log s)^{O(d)}$ over $\mathbb{F}_2$ such that

For all $x \in \{0,1\}^n$, $\Pr_{p \sim D(C)}[p(x) = C(x)] > \frac{3}{4}$.

**Proof Sketch:** Represent each gate by a polynomial.

**NOT gate:** $\text{NOT}(x_i) = 1 + x_i$

**XOR gate:** $\text{XOR}(x_1, \ldots, x_n) = \sum_i x_i \mod 2$.

**OR gate:** For all $x \in \{0,1\}^n$, observe that 

$$\Pr_{r \in \{0,1\}^n}[\text{OR}(x_1, \ldots, x_n) = \sum_i r_i x_i \mod 2] \geq \frac{1}{2}$$

Pick $R \in \mathbb{F}_2^{k \times n}$ at random, where $k$ = error parameter.

For all $x \in \{0,1\}^n$, 

$$\Pr_R[\text{OR}(x_1, \ldots, x_n) = 1 + \prod_j (1 + \sum_i R_{j,i} x_i) \mod 2] \geq 1 - \frac{1}{2^k}$$

This is a degree-$k$ polynomial simulating OR with error $< 1/2^k$.

Set $k = 10 \log(s)$ and replace all gates with the above polynomials.
Lower Bounds for AC0[⊕]

**Theorem [Razborov-Smolensky’87]** For every AC0[⊕] circuit \( C \) with \( n \) inputs, size \( s \), and depth \( d \), there is an efficiently samplable distribution \( D(C) \) of polynomials of degree \( (\log s)^{O(d)} \) over \( \mathbb{F}_2 \) such that

For all \( x \in \{0, 1\}^n \),

\[
\Pr_{p \sim D(C)} [p(x) = C(x)] > \frac{3}{4}.
\]

There are **probabilistic polynomials** of degree \( (\log s)^{O(d)} \) for AC0 [⊕]

**Corollary** For every AC0[⊕] circuit \( C \) with \( n \) inputs, size \( s \), and depth \( d \), there exists a single polynomial \( p(x_1, ..., x_n) \) of degree \( (\log s)^{O(d)} \) that agrees with \( C \) on \( \geq \frac{3}{4} \) of the points in \( \{0, 1\}^n \).

**Theorem** No polynomial of degree \( o(\sqrt{n}) \) agrees with MAJORITY on at least \( \frac{3}{4} \) of the points in \( \{0, 1\}^n \).

Therefore if an AC0[⊕] circuit \( C \) with \( n \) inputs, size \( s \), and depth \( d \) computes MAJORITY then \( (\log s)^{O(d)} \geq \sqrt{n} \).

E.g., \( O(1) \)-depth circuits require **exponentially** many gates for MAJORITY.
How can Algorithmists use this?

**Theorem [Razborov-Smolensky’87]** For every $\text{AC0}[\oplus]$ circuit $C$ with $n$ inputs, size $s$, and depth $d$, there is an efficiently samplable distribution $D(C)$ of polynomials of degree $(\log s)^{O(d)}$ over $\mathbb{F}_2$ such that

For all $x \in \{0, 1\}^n$, $\Pr_{p \sim D(C)} [p(x) = C(x)] > \frac{3}{4}$.

There are probabilistic polynomials of degree $(\log s)^{O(d)}$ for $\text{AC0} [\oplus]$

**Multipoint evaluation of polynomials (MM/FFT)**

$\Rightarrow$ Faster randomized algorithms!
Outline

• The Polynomial Method and Circuit Lower Bounds
• All-Pairs Shortest Paths in Dense Graphs
• Other Algorithmic Applications
• Future Work
All-Pairs Shortest Paths (APSP)

**Given:** \( n \)-node graph \( G \) defined by a weight function \( w : [n] \times [n] \rightarrow \mathbb{R}^+ \cup \{\infty\} \)

**Compute:** For all \( i, j \in [n] \), a shortest path from \( i \) to \( j \)

**Meaning?** *(Could take \( \Omega(n^3) \) space just to write down paths)*
All-Pairs Shortest Paths (APSP)

**APSP**

**Given:** $n$-node graph $G$ defined by a weight function $w : [n] \times [n] \to \mathbb{R}^+ \cup \{\infty\}$

**Compute:** $n$ by $n$ matrix $S$ with the property:
for all $i, j \in [n]$, $S[i,j] = k \in [n]$ such that there is a shortest path from $i$ to $j$ starting with edge $(i,k)$

Such an $S$ always exists, and can be described in $\Theta(n^2 \log n)$ bits. Given $S$, we can compute the shortest path between any pair $s$ and $t$ in $O(p)$ time, where $p$ is the number of edges in the path.
The Computational Model

We’re working with real weights, and an adjacency matrix

An instance of APSP is specified with $n^2$ real-valued registers. This model is designed to be as general as possible, such that Floyd-Warshall (1961) takes $O(n^3)$ steps to solve APSP.

The Real RAM Model

Random access machine model with two types of registers

<table>
<thead>
<tr>
<th>Register Types</th>
<th>Real-Valued</th>
<th>Typical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holds:</td>
<td>real numbers</td>
<td>$O(\log n)$ bits</td>
</tr>
<tr>
<td>Operations:</td>
<td>Addition, Subtraction, Comparison ($a \leq b$)</td>
<td>All the usual bit instructions</td>
</tr>
</tbody>
</table>

The outcome of a Comparison goes into a typical register
The Necessity of Cubic Time for APSP?

Major Open Question (in the 60’s):
Is $O(n^3)$ time optimal for APSP?

(partial) Answer [Kerr’71]
YES... if only additions/comparisons on reals are allowed

But the Real RAM allows us to do arbitrary bit operations on the outcomes of comparisons... could this help?

(better) Answer: [Fredman’75]
NO! On the Real RAM, APSP is in $\tilde{O}(n^3/\log^{1/3} n)$ time
($\tilde{O}$ omits poly$(\log \log n)$ factors)

This spawned a long line of work...
The Necessity of Cubic Time for APSP?

<table>
<thead>
<tr>
<th>Author</th>
<th>$\tilde{O}$ Runtime</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floyd, Warshall</td>
<td>$n^3$</td>
<td>1962</td>
</tr>
<tr>
<td>Fredman</td>
<td>$n^3 / \log^{1/3} n$</td>
<td>1975</td>
</tr>
<tr>
<td>Takaoka, Dobosiewicz</td>
<td>$n^3 / \log^{1/2} n$</td>
<td>1990,1991</td>
</tr>
<tr>
<td>Han</td>
<td>$n^3 / \log^{5/7} n$</td>
<td>2004</td>
</tr>
<tr>
<td>Takaoka, Zwick, Chan</td>
<td>$n^3 / \log n$</td>
<td>2004,2005</td>
</tr>
<tr>
<td>Han</td>
<td>$n^3 / \log^{5/4} n$</td>
<td>2006</td>
</tr>
<tr>
<td>Chan, Han-Takaoka</td>
<td>$n^3 / \log^2 n$</td>
<td>2007,2012</td>
</tr>
</tbody>
</table>

BIG QUESTION: Is APSP in $n^3 / \log^c n$ time, for every $c$?
Faster APSP (Via Circuit Complexity)

**Theorem 1:** There is a randomized algorithm for APSP running in \( \frac{n^3}{2L(n)} \) time, where \( L(n) \geq \Omega(\log n)^{1/2} \).

**Theorem 2:** There is some \( \delta > 0 \) and a deterministic algorithm for APSP running in \( \frac{n^3}{2L(n)} \) time, where \( L(n) \geq \Omega(\log n)^\delta \).

**Corollary:** APSP is in \( \frac{n^3}{\log^c n} \) time for every \( c \geq 1 \).

**Corollary:** Many other weighted graph problems now have similar running times as well!
Intuition

There is a natural matrix product associated with APSP

**Theorem** [Fischer-Meyer, Munro ‘71]
To solve APSP on n-node graphs in time $O(T(n))$, it suffices to compute the *min-plus matrix product* of $A, B \in \mathbb{R}^{n \times n}$

$$(A \circ B)[i, j] = \min_k (A[i, k] + B[k, j])$$

in $T(n)$ time.

(This fact is used in all algorithms for dense APSP.)
Intuition

Try to compute this matrix product via Razborov-Smolensky

Key Idea 1:

*Min-plus inner products* are EASY wrt circuit complexity!
They are computable with AC0 circuits

\[ \text{constant depth, AND/OR/NOT, polynomial size} \]

Key Idea 2:

EASY inner products can be reduced to polynomials over \( \mathbb{F}_2 \)

[Razborov-Smolensky’87] Randomized reduction from AC0
circuits to polylog-degree polynomials over \( \mathbb{F}_2 \): for every input,
the probability the polynomial agrees with the circuit is > ¾.

Key Idea 3:

Polynomials can be eff. evaluated on many pairs of points
[Coppersmith’82] (Very) fast rectangular matrix multiplication
1. Min-plus inner products are EASY

Let \( u, v \in \mathbb{N}^d \). Define \( (u \circ v) := \min_k (u[k] + v[k]) \)

Lemma: Let \( u_1, ..., u_n, v_1, ..., v_n \in \mathbb{N}^d \). For each \( i, j \), \( u_i \circ v_j \) can be computed with an \( (d \log n)^{O(1)} \)-size AC0 circuit, after \( \tilde{O}(n d^2) \) preprocessing.

Proof Sketch: Let \( \ell \) range from \( 1, ..., \log d \). Fix an \( i \) and \( j \).

Want to compute the \( \ell \)th bit of the integer \( k \) such that \( u_i[k] + v_j[k] \) is minimized (WLOG, this \( k \) is unique, depending only on \( i \) and \( j \)).

\[
\bigvee_{k' = 1, \ldots, d : \ell \text{th bit of } k' \text{ is 1}} \left[ u_i[k'] + v_j[k'] \text{ is minimum} \right]
\]

Pre-compute these diffs, and sort them! (Fredman’s trick)

\[
\bigvee_{k' = 1, \ldots, d : \ell \text{th bit of } k' \text{ is 1}} \bigwedge_{k''} \left[ u_i[k'] + v_j[k'] \leq u_i[k''] + v_j[k''] \right]
\]

\[
\bigvee_{k' = 1, \ldots, d : \ell \text{th bit of } k' \text{ is 1}} \bigwedge_{k''} \left[ u_i[k'] - u_i[k''] \leq v_j[k''] - v_j[k'] \right]
\]
1. Min-plus inner products are EASY

Let $u, v \in \mathbb{N}^d$. Define $(u \circ v) := \min_k (u[k] + v[k])$

**Lemma:** Let $u_1, \ldots, u_n, v_1, \ldots, v_n \in \mathbb{N}^d$. For each $i, j$, $(u_i \circ v_j)$ can be computed with an $(d \log n)^O(1)$-size AC0 circuit, after $\tilde{O}(n d^2)$ preprocessing.

For all $(k', k'')$ and $i = 1, \ldots, n$, sort the numbers $u_i[k'] - u_i[k''], v_i[k''] - v_i[k']$

Then, we can assume all numbers are in the range $\{1, \ldots, 2nd^2\}$.
Computing $(x \leq y)$ for two $(\log n)$-bit numbers is in AC0 of $(\log n)^O(1)$ size:

$$x \leq y \iff \left( \bigwedge_{i=1}^{t} (1+x_i+y_i) \right) \vee \bigvee_{i=1}^{t} \left( (1+x_i) \wedge y_i \wedge \bigwedge_{j=1}^{i-1} (1+x_j+y_j) \right)$$
Theorem [Razborov-Smolensky’87] For every $AC0[\oplus]$ circuit $C$ with $n$ inputs, size $s$, and depth $d$, there is an efficiently sampleable distribution $D(C)$ of polynomials of degree $O(\log s)^{O(d)}$ over $\mathbb{F}_2$ such that

For all $x \in \{0, 1\}^n$, \( \Pr_{p \sim D(C)}[p(x) = C(x)] > \frac{3}{4}. \)

3: Polynomial evaluation

Want to evaluate min-plus inner product on $n^2$ pairs of vectors.

We translate that problem into evaluating a multivariate polynomial at $n^2$ pairs of points. We can do this very efficiently, provided the polynomial is sparse.
3: Polynomial evaluation

Want to evaluate min-plus inner product on $n^2$ pairs of vectors.

We will translate that problem into evaluating a multivariate polynomial at $n^2$ pairs of points.

We can do this very efficiently, provided the polynomial is sparse.

Key Theorem: Given $A, B \subseteq \{0, 1\}^m$, $|A| = |B| = n$, and a polynomial $q(x_1, \ldots, x_m, y_1, \ldots, y_m)$ over $\mathbb{F}_2$, with $|q| \leq n^{0.1}$, we can evaluate $q$ on all $(x, y) \in A \times B$ in $n^2 \text{poly}(\log n)$ time.

Proof Idea: Embed the evaluation problem into an efficient rectangular matrix multiplication.
Towards an APSP Algorithm

Given: $A, B \in \mathbb{N}^{n \times n}$
Want to compute: $(A \circ B)[i,j] = \min_k (A[i,k] + B[k,j])$

Theorem [Folklore] Suppose we can multiply $n \times d$ and $d \times n$ matrices in $O(n^2 \cdot \text{polylog } n)$ time.
Then we can multiply $n \times n$ matrices in $O\left(\frac{n^3}{d} \text{polylog } n\right)$ time.

[Very simple: Just break the $n \times n$ matrices into $n/d$ multiplications of $n \times d$ and $d \times n$]

Now: We wish to maximize $d$ such that $n \times d$ and $d \times n$ min-plus matrix multiplication is in $n^2 \text{ poly}(\log n)$ time.
We will manage to have $d = 2^{(\log n)^\delta}$, for some $\delta > 0$
Idea: Replace min-plus inner product w/ polynomial over GF(2)
**Now:** Maximize $d$ such that $n \times d$ and $d \times n$ min-plus matrix product is computable in $n^2 \text{poly} (\log n)$ time.

Then: APSP will be in $\frac{n^3}{d} \text{poly} (\log n)$ time.

Given $A, B$ which are $n \times d$ and $d \times n$

Let $C$ be our AC0 circuit for the min-plus inner product of two $d$-length vectors with entries from $\mathbb{N}$ (after $\tilde{O}(nd^2)$ preprocessing)$\quad C$ has $O(d^2 \log n)$ inputs, $O(d \log d)$ outputs, $(d \log n)^{O(1)}$ size

For every output bit $j = 1, \ldots, O(\log d)$ of $C$,

a. Sample random $p^j_1, \ldots, p^j_{10 \log n} \sim D(C)$ simulating the circuit $C$

For all $i,j$ we have $\text{deg}(p^j_i) \leq (\log (d \log n))^c$;

hence the number of monomials $|p^j_i| \leq (d \log n)^{(\log(d \log n))^c}$

b. Evaluate $p^j_1, \ldots, p^j_{10 \log n}$ on all rows of $A$ and columns of $B$.

c. Output the majority answer. (Correct whp in all $n^2$ entries, by Chernoff)

Step b can be performed in $n^2 \text{poly} (\log n)$ time, provided $\quad |p^j_i| \leq (d \log n)^{(\log(d \log n))^c} \leq n^{0.1}$

Set $d = 2^{(\log n)^{1/(c+1)}/10} \rightarrow$ APSP in $n^3/2^{\Omega(\log n)^{1/(c+1)}}$ time
Outline

- The Polynomial Method and Circuit Lower Bounds
- All-Pairs Shortest Paths in Dense Graphs
- Other Algorithmic Applications
- Future Work
Other Algorithmic Applications

[W STOC’14] 0-1 Integer LPs with $n$ vars & $poly(n)$ constraints: $2^{n - n/polylog(n)}$ time
   Combination of probabilistic polynomials + sorting/preprocessing

[AWY SODA’15] Finding orthogonal pair among $n$ vectors in $\{0, 1\}^{c\log n}$: $n^{2 - 1/log(c)}$ time
   Computing partial match queries in batch, evaluating a CNF in batch
   Carefully count monomials in probabilistic polys for these functions

[AW FOCS’15] Closest Pair in Hamming metric in $c\log n$ dimensions: $n^{2 - \frac{1}{c^2\log(c)}}$ time
   Develop new probabilistic polynomial for MAJORITY function

[CW SODA’16] Deterministic APSP: $n^{3 - \frac{1}{\sqrt{\log(n)}}}$ time
   Deterministic counting of $k$-SAT Solutions: $2^{n - \frac{n}{O(k)}}$ time
   Applies modulus-amplifying polynomials [BT’91] & small-bias sets [NN’91]

[LPTWY ????] Solve systems of degree-$d$ equations over $\mathbb{F}_q$: $q^{n - \frac{n}{O(d)}}$ time

[ACW ????] $(1 + \varepsilon)$-Approximate Closest Pair in L2 metric: about $n^{2 - \varepsilon^3}$ time
   New notion of “probabilistic polynomial threshold functions”
Future Work

• FIND MORE APPLICATIONS!

• Circuit complexity has developed many tools for analyzing and manipulating “weak” circuits. Often, these “weak” circuits capture worst-case algorithms that we believe to be best-possible. When can these tools be applied to improve algorithms?

• Exploit the fact that we aren’t interested in proving circuit lower bounds, just want to evaluate circuits! E.g. Develop probabilistic notions of matrix rank, polynomial threshold functions, ... whatever works!
Thank you!